US006065008A

United States Patent [19]

Simon et al.

[11] Patent Number: 6,065,008

[45] Date of Patent: May 16, 2000

[54] **SYSTEM AND METHOD FOR SECURE FONT SUBSET DISTRIBUTION**

[75] Inventors: **Daniel R. Simon**; **Josh Benaloh**; **Donald D. Chinn**, all of Redmond; **Gregory Hitchcock**, Woodinville; **David Meltzer**, Seattle, all of Wash.

[73] Assignee: **Microsoft Corporation**, Redmond, Wash.

[21] Appl. No.: **08/942,036**

[22] Filed: **Oct. 1, 1997**

[51] **Int. Cl.**[7] **G06F 17/30**

[52] **U.S. Cl.** **707/10**; 707/9; 707/529; 707/542; 345/467; 380/3; 380/4

[58] **Field of Search** 707/542, 908, 707/10, 1, 9, 529; 345/467; 380/3, 4

[56] **References Cited**

U.S. PATENT DOCUMENTS

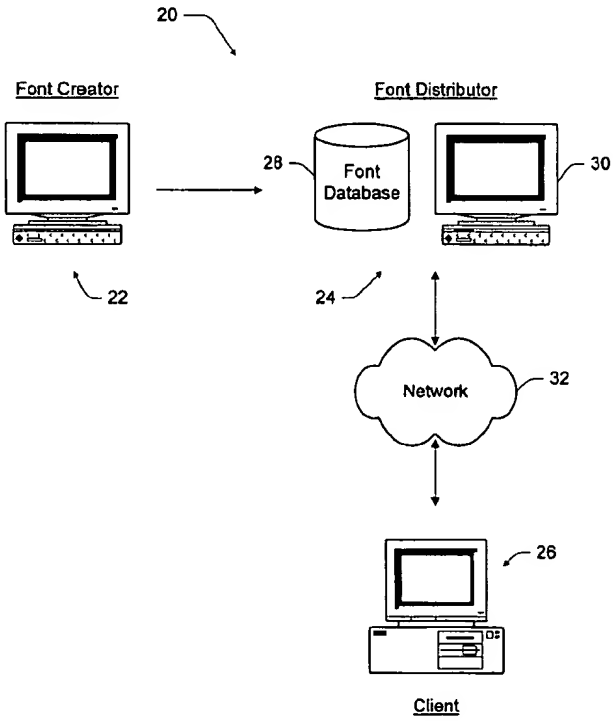
5,528,742	6/1996	Moore et al.	707/542
5,737,599	4/1998	Rowe et al.	707/104
5,859,648	1/1999	Moore et al.	345/471
5,877,776	3/1999	Beaman et al.	345/472
5,893,915	4/1999	Cordell et al.	707/513
5,940,581	8/1999	Lipton	358/1.11
5,990,907	11/1999	Colletti	345/467

Primary Examiner—Anton W. Fetting
Assistant Examiner—Greta L. Robinson
Attorney, Agent, or Firm—Lee & Hayes, PLLC

[57] **ABSTRACT**

This invention concerns a system and method for securely distributing subsetted fonts from a distributor to a client. The system includes a signing module to construct an authentication tree having leaves formed of glyphs, one or more intermediate levels of nodes computed as one-way functions of the glyphs, and a root computed as a one-way function of the nodes. The signing module digitally signs the root of the authentication tree using a private signing key unique to the font creator or distributor. The system has a subsetting module to construct a font subset file that contains selected glyphs and other data to be included in a font subset. The font subset file also holds the digitally signed root of the font authentication tree and one or more authentication values of the authentication tree that represents non-selected glyphs and data of the font that are not contained in the font subset. The font subset file is distributed to requesting clients. An authentication module at the client authenticates the font subset file received from the distributor. The authentication module reconstructs the root of the authentication tree using the selected glyphs and data in the font subset and the authentication values that represent the non-selected glyphs and data not contained in the font subset. The authentication module also produces an unsigned version of the digitally signed root using a public key of the font creator to produce an unsigned root digest. The authentication module compares the unsigned root to the reconstructed root and if and only if they match, authenticates the font subset file as originating from the distributor (or font creator) and not being subsequently altered.

27 Claims, 8 Drawing Sheets



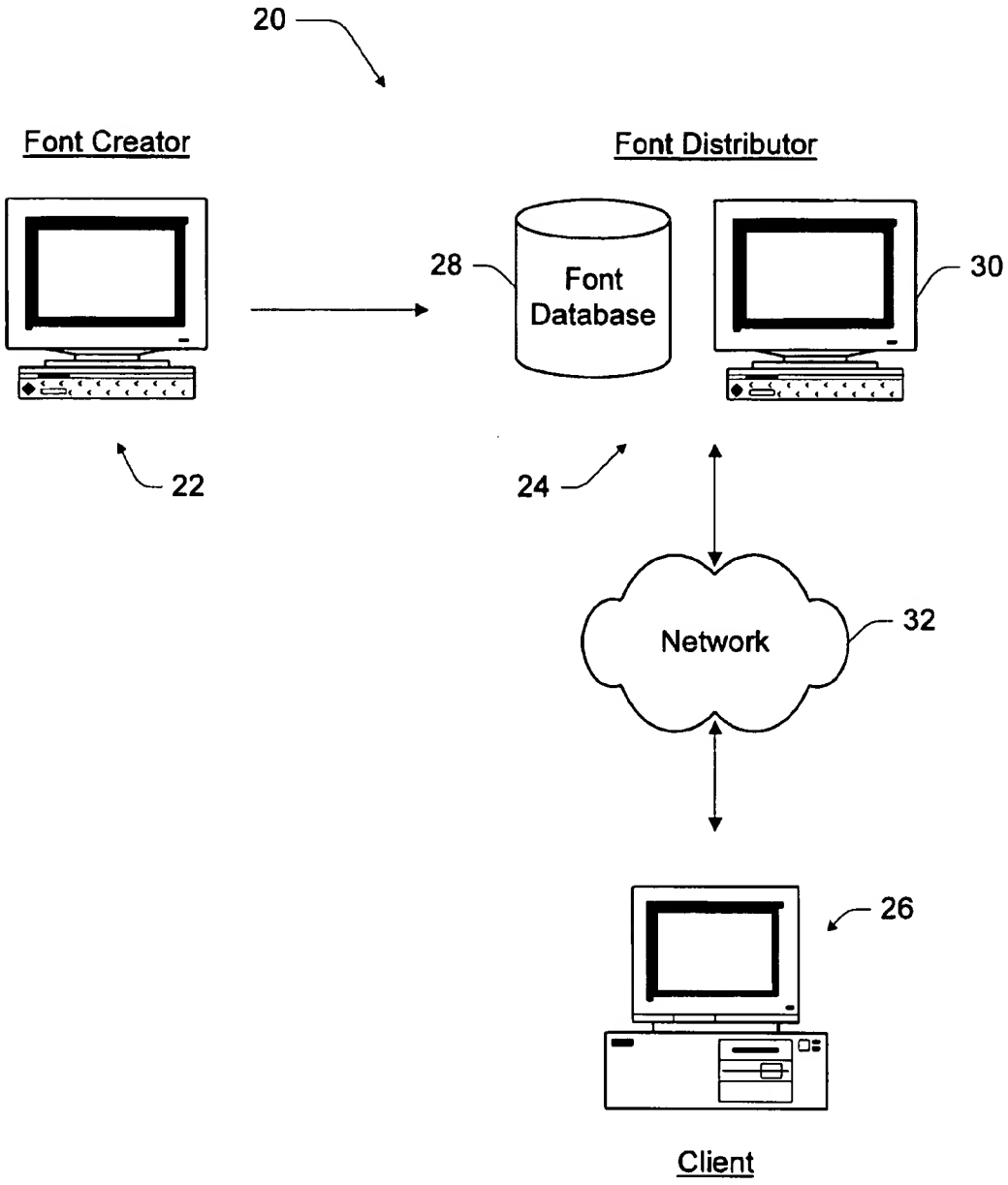

20
Font Creator
Font Distributor
28
Font Database
30
22
24
Network
32
26
Client

Fig. 1

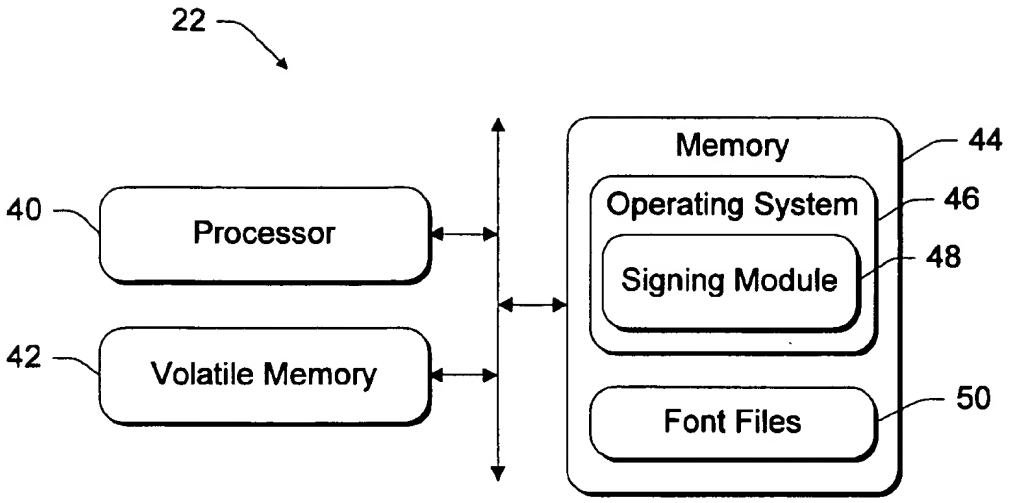

22
40
Processor
42
Volatile Memory
Memory
44
Operating System
46
Signing Module
48
Font Files
50

Fig. 2

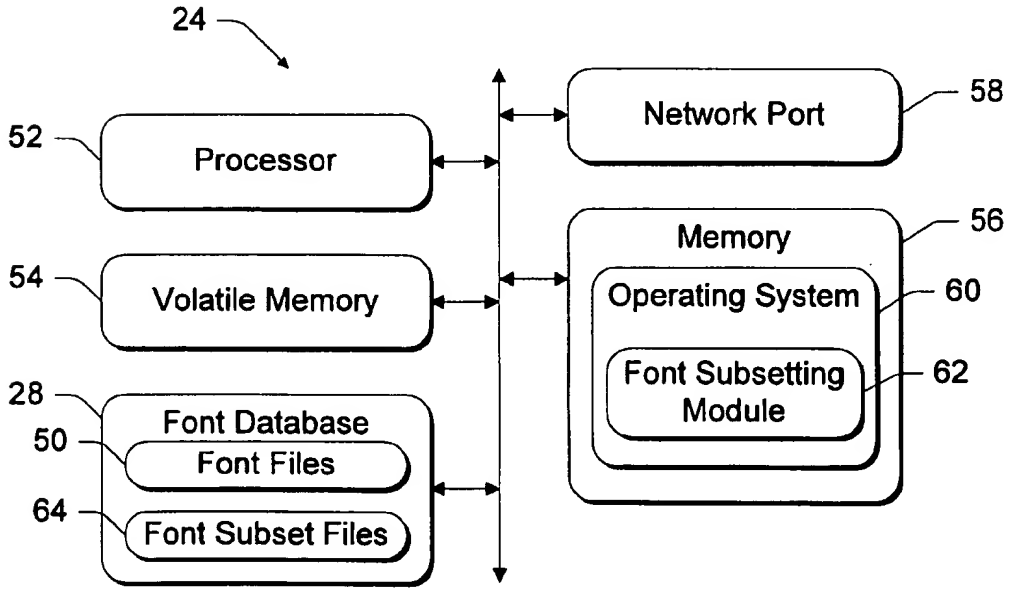

24
52
Processor
54
Volatile Memory
28
Font Database
50
Font Files
64
Font Subset Files
Network Port
58
Memory
56
Operating System
60
Font Subsetting Module
62

Fig. 3

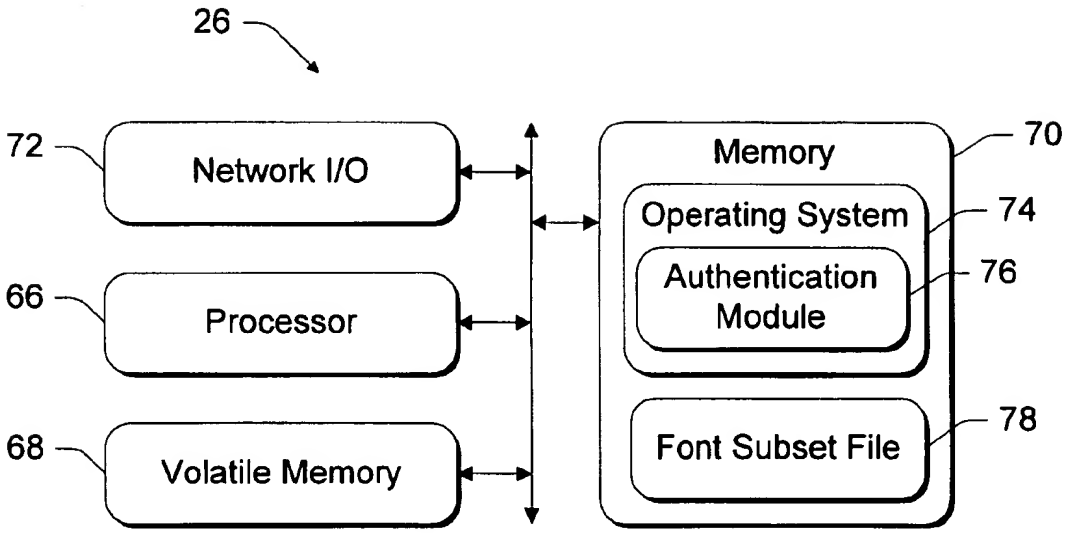
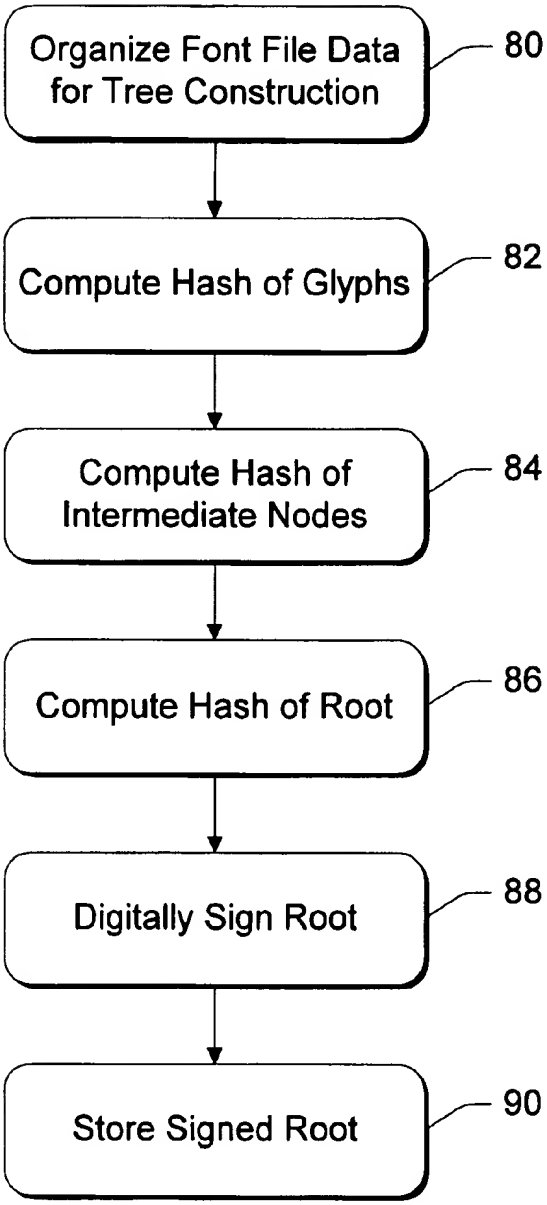

26
72
Network I/O
66
Processor
68
Volatile Memory
Memory
70
Operating System
74
Authentication Module
76
Font Subset File
78

Fig. 4



Tree Construction Process

Fig. 5

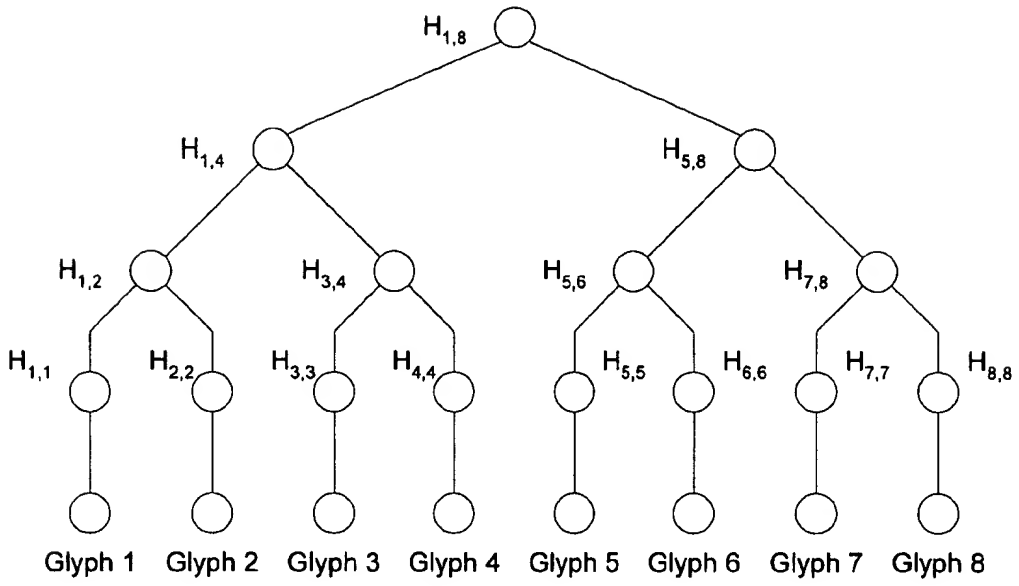

$H_{1,8}$
$H_{1,4}$
$H_{5,8}$
$H_{1,2}$
$H_{3,4}$
$H_{5,6}$
$H_{7,8}$
$H_{1,1}$
$H_{2,2}$
$H_{3,3}$
$H_{4,4}$
$H_{5,5}$
$H_{6,6}$
$H_{7,7}$
$H_{8,8}$
Glyph 1
Glyph 2
Glyph 3
Glyph 4
Glyph 5
Glyph 6
Glyph 7
Glyph 8

Fig. 6

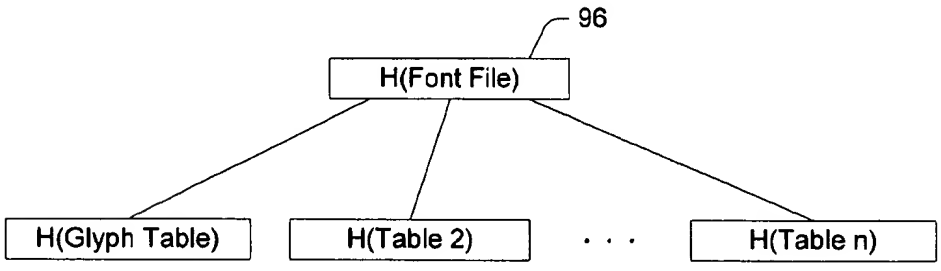
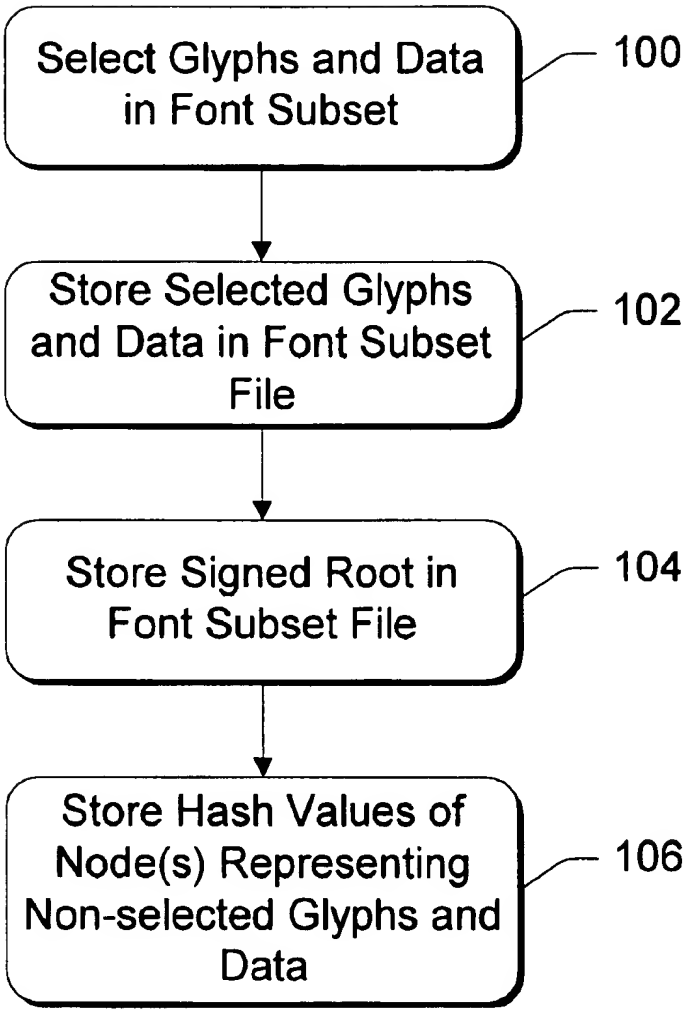

96
H(Font File)
H(Glyph Table)
H(Table 2)
. . .
H(Table n)

Fig. 7



Font Subsetting Process

Fig. 8

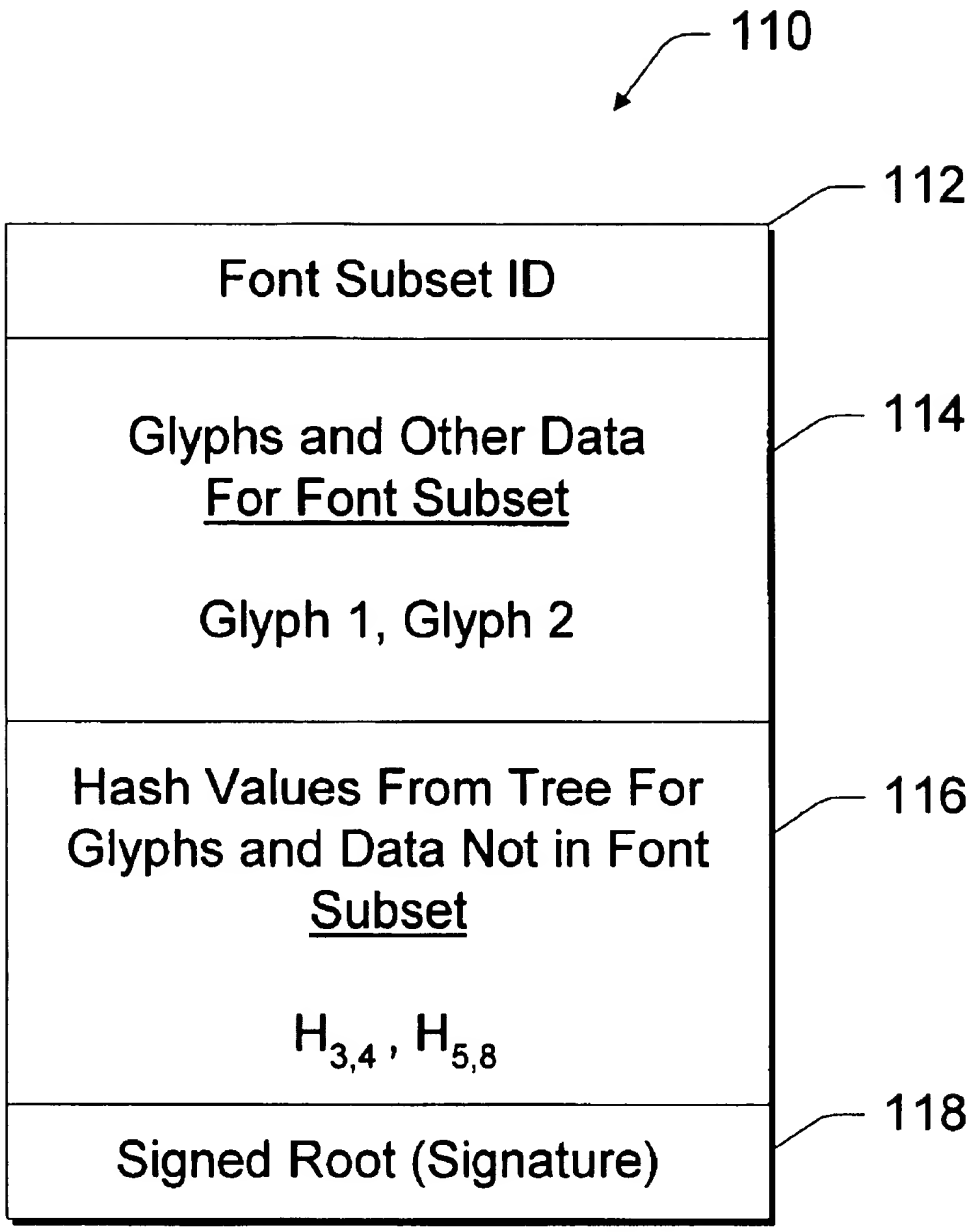
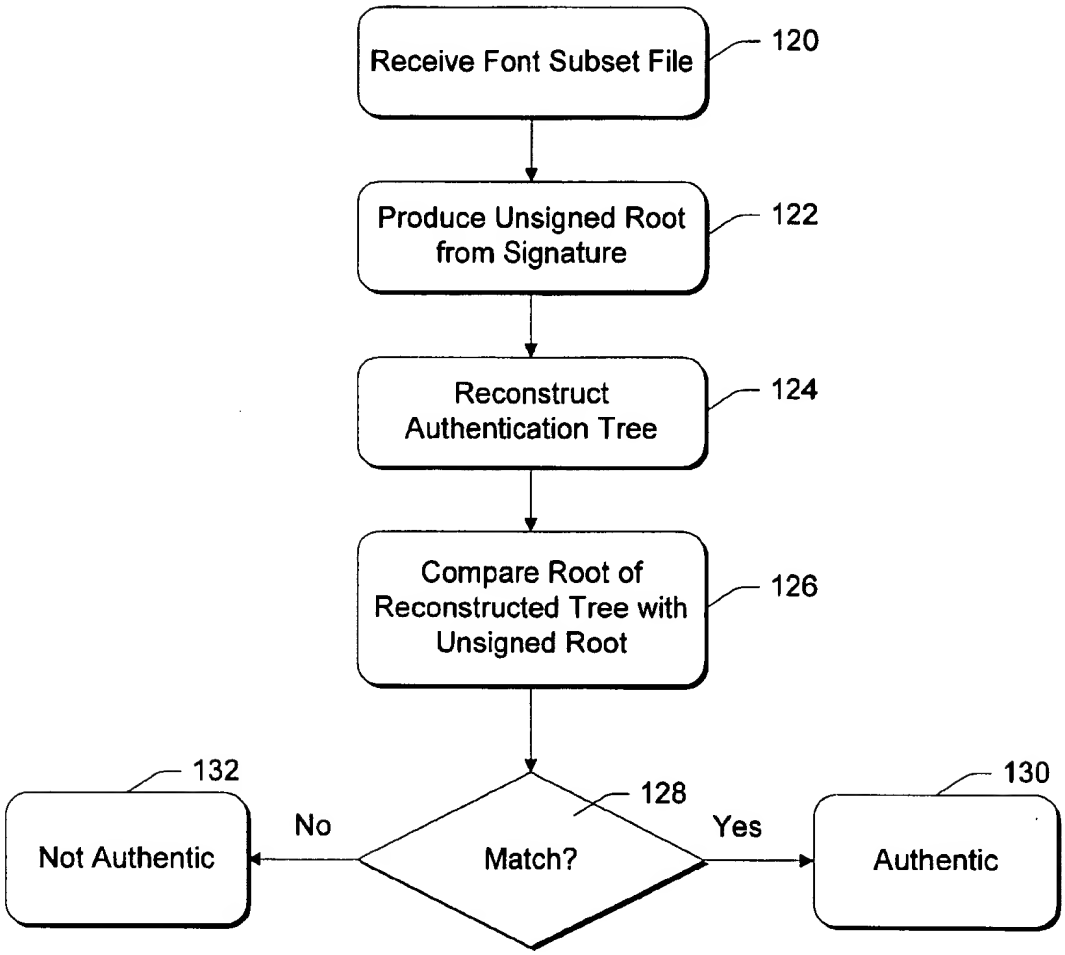

110
112
Font Subset ID
114
Glyphs and Other Data For Font Subset
Glyph 1, Glyph 2
116
Hash Values From Tree For Glyphs and Data Not in Font Subset
$H_{3,4}$, $H_{5,8}$
118
Signed Root (Signature)

Fig. 9



Authentication Process

Fig. 10

SYSTEM AND METHOD FOR SECURE FONT SUBSET DISTRIBUTION

TECHNICAL FIELD

This invention relates to systems and methods for electronically distributing font subsets. More particularly, this invention relates to systems and methods for electronically distributing a font subset or a piece of media, which is extracted from a larger digitally signed font or media, in a secure manner that permits a recipient to authenticate the source of the font subset or media piece.

BACKGROUND OF THE INVENTION

Fonts are used by computers for on-screen displays and by printers for hard-copy printout. There are many different kinds of fonts available today. A "font" is a set of characters of the same typeface (e.g., Times New Roman, Arial, and Courier), style (e.g., italics), stroke weight (e.g., bold), and size (e.g., 12 point). More generally, a font defines how a set of characters appears when displayed or printed.

A font is commonly defined by a set of mathematical rules and glyph information contained in a font file. The rules define how a computer monitor or printer converts data into pixel data that is specific to turning on and off the appropriate dots, or pixels, to form the glyphs. A "glyph" is an exact shape of a character form. For instance, in a cursive (handwritten) font, the character represented by a lowercase "r" is rendered as one of two possible glyphs, depending on what character precedes it in the text.

To provide better quality glyphs, the amount of data contained in a font file continues to increase. As an example, a single font file for a Latin alphabet might consume 60 Kbytes of memory. For a Japanese character set, a single font file might be 15 to 17 Mbytes.

As the size of the font file grows, it is less convenient to supply the entire font for each application or usage of the font. For example, when a user downloads a Web page from the Internet, there is often no need for an entire font to accompany the Web page for rendering the small amount of textual content expressed on the page. Rather than downloading the entire font file (which might be large), a subset of the font file is downloaded. The subsetted font contains enough rules and glyph information to present the characters contained in the Web page.

The practice of font subsetting is well-known. A fairly recent problem that affects font subsetting concerns security. In particular, fonts that are passed over networks, and particularly public networks such as the Internet, face several security risks. One risk is that a party will copy the font file and reproduce it without permission, or make minor alterations, and then republish the font file for use in situations not intended by the font designer. This is particularly troubling for fonts protected by intellectual property. Font designers often license fonts with express conditions of when a font can, or cannot, be transported with a document or application. An unscrupulous party might attempt to tamper with the font file to change these conditions so that the font appears to legally grant permission to transfer the file in a manner not intended by the designer.

Another problem is that a font file, like other data files and code, might be infected with a virus. Still another problem is that an imposter might publish a font file of inferior quality, or one that is tainted, under the name of a reputable font designer. For these problems, it would be beneficial if the recipient of a font file could authenticate the source of the font file, and determine whether the font file has been tampered with since leaving the source.

To address these problems, font designers or publishers can digitally sign their font files by treating the file as a single stream of bytes. This technique is called a "flat file" approach. A recipient of a file can use the digital signature to verify that the font file was created by the designer and has not been subsequently altered. Unfortunately, the digital signature applies only to the original stream of bytes, and so when a font is subsetted, the signature is of no use for the subsetted font. By definition, the subsetting process necessarily converts the font file to a subsetted font file, and the digital signature is no longer valid for the subsetted font file.

One solution to the subsetting problem is to create, sign, and publish every possible subset of a font file. This solution is impractical because the number of subsets that are possible for a font is large.

Another solution is for the designer to predetermine a few common subsets, such as the most likely script/language ranges, and include signatures for each subsetted version of the font. The multiple signatures could then be stored in the font file. This technique is limited, however, in that the designer could not begin to anticipate all of the subsetting combinations that a content producer might desire. Moreover, the content producer might be unduly restricted to applying only the predetermined subsetted files.

Another possible solution is to establish a font server which, when a subset of a font is requested by a user, creates and signs the subsetted font on-the-fly and then issues the font to the user. In this scenario, the font servers would belong to font creators and would have access to private signing capabilities associated with the font creators, so that the font servers could create new signatures on-the-fly. With this technique, however, a tremendous investment in infrastructure is required. Additionally, this approach would potentially expose the signer's private key. It also requires content publishers to contact a font vendor every time a different subset is needed, which can be quite burdensome.

Accordingly, there is a need to develop a technique other than a flat file approach for subsetting a font in a manner that enables a recipient to verify the authenticity of the subsetted font.

SUMMARY OF THE INVENTION

This invention concerns a system and method for securely distributing subsetted fonts. The system includes a signing module, a subsetting module, and an authentication module. The signing and subsetting modules are resident at the font designer (or distributor), and the authentication module is resident at the client who requests and receives the subsetted font.

For a given font, the signing module constructs an authentication tree having leaves formed of glyphs, one or more intermediate levels of nodes computed as one-way functions of the glyphs, and a root computed as a one-way function of the nodes. In one implementation, the one-way function is a hash function. In addition to glyphs, other data from the font file (e.g., permissions) might also be included in the authentication tree. The signing module digitally signs the root of the authentication tree using a private key unique to the font creator or distributor.

The subsetting module subsets the font to form a font subset requested by a client. The subsetting module constructs a font subset file that contains glyphs and other data to be included in the font subset. The font subset file also holds the digitally signed root of the font authentication tree

and one or more authentication values of the authentication tree that represents glyphs and data of the font that are not contained in the font subset. The font subset file is then distributed to the client.

At the client, the authentication module authenticates the font subset file received from the distributor. The authentication module reconstructs the root of the authentication tree using the glyphs and data in the font subset and the authentication values that represent glyphs and data not contained in the font subset. The authentication module also produces an unsigned root by using the public key of the font creator. The authentication module compares the unsigned root to the reconstructed root and if and only if they match, authenticates the font subset file as originating from the distributor and not being subsequently altered.

BRIEF DESCRIPTION OF THE DRAWINGS

The same reference numbers are used throughout the drawings to reference like components and features.

FIG. **1** is a diagrammatic illustration of a server-client architecture.

FIG. **2** is a block diagram of the server at a font creator.

FIG. **3** is a block diagram of a server at a font distributor.

FIG. **4** is a block diagram of the client.

FIG. **5** is a flow diagram of a method for constructing an authentication tree for a font file.

FIG. **6** is a diagrammatic illustration of an authentication tree for eight glyphs in a font file.

FIG. **7** is a diagrammatic illustration of a primary root portion of the authentication tree for the font file.

FIG. **8** is a flow diagram of a method for subsetting a font file.

FIG. **9** is a diagrammatic illustration of a data structure for a subsetted font file.

FIG. **10** is a flow diagram of a method for authenticating a subsetted font file.

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENT

The following discussion assumes that the reader is familiar with cryptography. For a basic introduction of cryptography, the reader is directed to a text written by Bruce Schneier and entitled "Applied Cryptography: Protocols, Algorithms, and Source Code in C," published by John Wiley & Sons with copyright 1994 (with a second edition in 1996), which is hereby incorporated by reference.

FIG. 1 shows a computer network system **20** having three participants: a font creator **22**, a font distributor **24**, and a client **26**. The font creator **22** is illustrated with a computer server to develop digitally signed font files that can be delivered to and used by the font distributor. The font creator **22** develops a font file in such a manner that the creator's digital signature is applied to the whole font file, and also to subsets of the font file. The signed font file is sent to the distributor in a number of conventional means, such as over a network, on a diskette, CD-ROM or other computer-readable medium, and the like. One example of a font creator is a company that develops computer-generated fonts, such as Adobe.

The font distributor **24** has a font database **28** to store the digitally signed font files produced by the font creator. The font distributor **24** also has a computer **30** for serving content over a network **32** to the client **26**. The distributor server **30** is configured to distribute the entire signed font file or signed subsets of the font files. The distributor server **30** is configured to subset the font file in such a manner that the client **26** can authenticate the font subset file as being created by the font creator **22** and verify that the font subset file has not been altered. Additionally, the subset file may contain information to permit the client **26** to authenticate that the font distributor **24** distributed the file. An example of a font distributor is a Web page owner who might wish to subset the font file for convenience of the client.

The network **32** can be implemented in a variety of ways. For instance, the network **32** might be a wireless network, such as satellite, radio, microwave, and so forth. The network **32** might also be a wire-based network, such as the Internet, a LAN (local area network), or a WAN (wide area network).

FIG. **2** shows the font creator server **22** in more detail. It includes a processor **40**, a volatile memory **42** (e.g., RAM), and a non-volatile memory **44** (e.g., ROM, hard disk drive, floppy disk drive, CD-ROM, etc.). The server **22** runs an operating system **46**, such as the Windows NT server operating system from Microsoft Corporation or a UNIX-based operating system. The operating system **46** includes a signing module **48**. This module is implemented in software and can be embodied in different forms, such as routines within the operating system, or dynamically linked libraries (DLLs), or interfaces, or the like. The signing module **48** is invoked to digitally sign the font files. More particularly, the signing module **48** is configured to construct an authentication tree for a font and to digitally sign the root of the authentication tree. The signing module **48** stores the digital signature together with the font in a font file **50**. The font file **50** can then be supplied to the font distributor for publication or subsetting.

FIG. **3** shows the font distributor server **30** in more detail. It includes a processor **52**, a volatile memory **54** (e.g., RAM), a non-volatile memory **56** (e.g., ROM, hard disk drive, floppy disk drive, CD-ROM, etc.), and font database **28**. The server **30** also has a network port **58** that provides access to the network **32**. Depending on the configuration of network **32**, the network port **58** might be implemented as a wireless transmitter, a modem, an Internet connection, and the like.

The server **30** runs an operating system **60**, such as the Windows NT server operating system from Microsoft Corporation or a UNIX-based operating system. The operating system **60** includes a subsetting module **62**. This module is implemented in software and can be embodied in different forms, such as routines within the operating system, or dynamically linked libraries (DLLs), or interfaces, or the like.

The subsetting module **62** is invoked to subset a font to produce a font subset. The subsetting module **62** constructs, from a font file **50**, a font subset file **64** containing the font subset, the digitally signed root of the authentication tree, and one or more authentication values of the authentication tree that represent portions of the font that are not contained in the font subset. The font subset file **64** may also be stored in font database **28**.

In the illustrated implementation, the functions of font creation and subsetting are performed by different entities. The signing module **48** resides at a font creator to construct a font file **50** that contains the creator's signature. The subsetting module **62** resides at a distributor to construct the font subset file **64** from the font file **50**. In another implementation, the functions of font creation and font subsetting are performed by the same entity. In this case, the

signing module **48** and font subsetting module **62** are incorporated into the same computing unit at the entity. From the perspective of the client, the font subset file is issued by a source that may comprise one or both of the signing and subsetting modules.

FIG. **4** shows the client **26** in more detail. It has a central processing unit comprising a processor **66**, a volatile memory **68** (e.g., RAM), and a non-volatile memory **70** (e.g., ROM, hard disk drive, floppy disk drive, CD-ROM, etc.). The client **26** also has a network I/O (input/output) **72** for accessing the network **32**. The network I/O **72** can be implemented, for example, as a wireless receiver, a modem, a network connection, or the like.

The client **26** executes an operating system **74** that supports multiple applications. The operating system **74** is preferably a multitasking operating system that allows simultaneous execution of multiple applications in a graphical windowing environment. One preferred operating system is a Windows-brand operating system sold by Microsoft Corporation, such as Windows 95, Windows NT, Windows CE, or other Windows-derivative versions. It is noted, however, that other operating systems may be employed, such as the Macintosh operating system from Apple Computer, Inc.

An authentication module **76** is shown incorporated into the operating system **74**. The authentication module **76** is invoked to authenticate a font subset received from the distributor **24**. When a font subset file **78** is received from the server, it is stored in memory **70**. The authentication module **76** reconstructs the root of the authentication tree for the whole font using the font subset and the authentication values contained in the font subset file. The authentication module **76** authenticates the font subset if and only if the root from the font subset file matches the reconstructed root.

A method for securely distributing font subsets that can be authenticated by the client will now be described with reference to FIGS. **5–10**. According to one aspect of this invention, the font file is represented as a tree structure for purposes of applying a digital signature. The font file contains glyphs and other data used to construct a set of characters. The font file is typically organized as multiple tables that contain different information, including glyph outlines, usage restrictions, metrics, and so forth. Glyph data is the largest component of the font file.

FIG. **5** shows steps in a method for constructing an authentication tree for the font. The signing module **48** of the font creator's server operating system **46** performs the steps of FIG. **5**. At step **80**, the signing module **48** organizes the data from the font file as low level leaves from which an authentication tree can be constructed. For discussion purposes, the steps of FIG. **5** are described in the context of creating a portion of the authentication tree for the glyph portion of the font file, wherein individual glyphs are represented as leaves in the authentication tree.

FIG. **6** shows a portion of the authentication tree for glyphs **1–8** of the font file. Glyphs **1–8** are arranged as the leaves at the bottom of the tree.

At step **82** in FIG. **5**, the signing module **48** uses a hash function to compute a hash value for each glyph. A hash function is a one-way mathematical function that converts an input data stream into a fixed-size, often smaller, output data stream that is representative of the input data stream. Hash values do not uniquely identify the large data streams, but it is assumed to be difficult to find two data streams that produce the same hash value; hence their effect is as if they were unique identifiers. The hash function is also deterministic in that two identical definitions passed through the same hash function will result in the same hash value. One example of a suitable hash function is SHA-1 (Secure Hash Algorithm).

In FIG. **6**, the hash value of glyph **1** is represented as $H_{1,1}$; the hash value of glyph **2** is represented as $H_{2,2}$; and so forth. It is noted, however, that other types of one-way functions besides hashing functions may be used to compute authentication values within the tree.

At step **84** in FIG. **5**, the signing module **48** computes a hash value for each intermediate node above the leaves. A node represents the conjunction of multiple hash values from a next lowest level in the tree. In this example, the authentication tree is binary, although non-binary trees may also be used. Each node branches to two nodes beneath it, thus representing 2^{level} leaves, where "level" is the number of levels from the node to the leaves. For example, each node at one level above the leaves represents two leaves (i.e., 2^1 leaves); each node at the next level up represents four leaf nodes (i.e., 2^2 leaves); and so forth.

Consider the node that combines glyphs **1** and **2**. The hash value of this node, $H_{1,2}$, is represented as follows:

$$H_{1,2}=F(H_{1,1}, H_{1,2})$$

where the function F is a one-way function that takes two values as input. In one implementation, the function F concatenates its inputs and applies a hash function (e.g., SHA-1). The node at the next level up represents four glyphs **1–4**. The hash value of this node, $H_{1,4}$, is represented as follows:

$$H_{1,4}=F(H_{1,2}, H_{3,4})=F(F(H_{1,1}, H_{2,2}, F(H_{3,3}, H_{4,4}))$$

The authentication tree has a root node at the very top. This root represents all of the underlying leaves. The root of the FIG. **6** tree portion represents all eight glyphs **1–8**. At step **86** in FIG. **5**, the signing module computes a hash value for the root of the tree. In this example, the hash value for the root is as follows:

$$H_{1,8}=F(H_{1,4}, H_{5,8})$$

In general, if there are n glyphs, then $H_{1,n}$ is the hash value representing all the glyphs.

Assuming there are only eight glyphs in the font file, the glyph branch of the authentication tree is now complete. As noted above, the glyph portion is only one component, albeit the predominate one, of the font file. Other tables are also found in the font file, such as a metrics table. The signing module **48** separately hashes each table and applies F to these hash values to compute a hash for the root, which represents the entire font file. Note that the hash values for these other tables might be computed using a flat file approach (if their contents does not change when the font file is subsetted), the technique of the authentication tree (if their data is organized similar to that of the glyph data), or a combination of these techniques.

FIG. **7** shows a root **96** for the authentication tree of the entire font file. This root is formed from concatenating the hash values of all roots of the sub-trees for all tables **1**-n of the font file, including the glyph table, and computing a hash value as follows:

$$H(\text{Font File})=F(H(\text{Glyph Table}), H(\text{Table 1}), \ldots, H(\text{Table n}))$$

where F is allowed to have multiple values as input.

At step **88** in FIG. **5**, the signing module **48** digitally signs the root using a digital signature unique to the font designer or distributor. One technique for forming this digital signature is to sign the hash value $H_{1,8}$ of the root using a private signing key from an asymmetric public/private key pair that is unique to the designer or distributor. The public/private keys form the foundation of public key cryptography systems. The keys are based upon a mathematical relationship in which one key cannot be calculated (at least in any reasonable amount of time) from the other key. The public key is distributed to other parties and the holder maintains the private key in confidence. Public key cryptography is well-known. An example of one well-known asymmetric cipher is the RSA cryptographic algorithm named for the creators Rivest, Shamir, and Adleman.

At step **90** in FIG. **5**, the signing module **48** stores the signed root as part of the font file in the font database.

One technique for constructing authentication trees is described in U.S. Pat. No. 4,309,569, which issued Jan. 5, 1982 in the name of Merkle. This patent is incorporated by reference.

A binary tree has been described for convenience. It is noted that the authentication tree can be organized in many different forms. The tree can be made flatter, whereby each node branches to more than two underlying nodes. Moreover, some optimization might be used to group glyphs or other data in a manner that would reduce the size of the tree or the number of authentication values needed to reconstruct the root for typical subsets.

FIG. **8** shows steps in a method for subsetting a font file that contains the digitally signed root. The subsetting module **62** of the font distributor's server operating system **60** performs the steps. At step **100**, the subsetting module **62** selects the glyphs and data to be included in the font subset. This data is obtained from the various tables in the font file. The subsetting module **62** stores the selected glyphs and data in a font subset file (step **102** in FIG. **8**). The subsetting module **62** also stores the digitally signed root in the font subset file (step **104** in FIG. **8**).

The subsetting module **62** further includes, as part of the font subset file, the authentication values of the authentication tree that represents portions of the font that are not contained in the font subset. In our example, the subsetting module **62** stores the hash values of the non-selected glyphs and data (step **106** in FIG. **8**). To improve space efficiency, a single hash value representing a group of contiguous missing glyphs or data can be included in the font subset file, rather than the hash value for each missing glyph or datum.

As an example, suppose the font subset of the glyph tree in FIG. **6** calls only for glyphs **1** and **2**. The subsetting module **62** constructs a font subset file that contains the subsetted glyphs **1** and **2**, the signed root, and the hash values for non-selected glyphs **3–8**. Rather than storing each individual hash value $H_{8,8}$, however, the non-selected glyphs **3** and **4** can be represented by the node hash value $H_{3,4}$ and the non-selected glyphs **5–8** can be represented by the node hash value $H_{5,8}$. FIG. **9** shows a data structure **110** for the font subset file. It has a subset identification field **112** that contains enough information to determine which glyphs are present in the font subset and a content field **114** that contains the selected glyphs and data for the font subset. In this example, the content field **114** holds the selected glyphs **1** and **2**. The content field **114** contains glyph-specific information, such as instructions for how to render a specific glyph. The field also contains font-wide information, such as the name of the font, its publisher, permissions, and general information for how to render glyphs of the font. The font subset data structure **110** also includes an authentication field **116** with one or more authentication values from the font authentication tree that represents all remaining glyphs and data from the font that are not part of the font subset. In this example, the authentication field **116** might hold the hash value representing glyphs **3** and **4** (i.e., the hash value $H_{3,4}$) and the hash value representing glyphs **5–8** (i.e., the hash value $H_{5,8}$). A signature field **118** is included in the font subset data structure **110** to hold the digitally signed root of the font authentication tree. Once constructed, the font subset file is stored in the memory of the server, and/or downloaded to the client.

It is noted that the subsetting module **62** can further subset an already subsetted font file, using similar techniques, to compute the appropriate values for the authentication field **116**. That is, given a subset S of present glyphs, authentication values for non-present glyphs, and a subset S' of S, one can compute the authentication values corresponding to the glyphs missing from S'. These values are the same values that would have been computed by the subsetter if it had been given as input the original font file and S'.

FIG. **10** shows steps in a method for authenticating a font subset file at the client. The authentication module **76** of the client operating system **74** performs the steps. At step **120**, the client computer receives the font subset file from the server, stores it in memory, and makes it available to the authentication module **76**. The authentication module **76** produces the unsigned root by applying the server's public key to the digitally signed root contained in the signature field **118** (step **122** in FIG. **10**).

At step **124** in FIG. **10**, the authentication module **76** uses the data glyphs from content field **114** and the authentication values from authentication field **116** to reconstruct the authentication tree for the whole font file. Starting with the glyphs in content field **114** (i.e., glyphs **1** and **2**), the authentication module **76** computes the hash values $H_{1,1}$, and $H_{2,2}$, then computes the hash value $H_{1,2}$ using F, and so forth.

When the portions of the tree containing non-selected glyphs are needed, the authentication module **76** uses the hash values recomputed from the glyph data together with the hash values representing the non-selected glyphs. For example, to produce the hash value $H_{1,4}$, the authentication module **76** applies F to the recomputed value $H_{1,2}$ and the value $H_{3,4}$ received in authentication field. Then, to compute the root hash digest $H_{1,8}$, the authentication module **76** applies F to the recomputed value $H_{1,4}$ and the value $H_{5,8}$ received in authentication field. At this point, the authentication module has recreated the hash digest for the root of the glyph portion of the authentication tree. Any other information contained in the font subset file pertaining to other tables is then used to compute the hash digest of the root for the entire authentication tree.

At step **126** in FIG. **10**, the authentication module **76** compares the reconstructed root with the unsigned root. If the two values match (i.e., the "yes" branch from step **128**), the authentication module **76** deems the font subset file authentic (step **130**). Conversely, if the two values fail to match (indicating that the file was not signed by the alleged source or has been tampered with), the authentication module **76** informs the operating system or user that the font subset file is not authentic (step **132**). If the font subset is not authentic, the operating system can make a determination whether to use the font subset file. The operating system may still decide to use the font subset file, but with the knowledge that it may risk utilizing a potentially tainted or infected file.

The system and method of securely subsetting a font file is advantageous in that it allows the distributor to supply any

combination of font subsets, each of which can be easily verified as belonging to the same source, without having to digitally sign each subsetted font. Another advantage is that the technique accommodates all font information in addition to glyphs, such as intellectual property permissions, so that a font file can be subsetted in essentially any permutation of glyphs, permissions, and other font data.

Another, less preferred, technique to solving the problems addressed in the Background section is to digitally sign every glyph or other piece of data in the font. When the font is subsetted, the requested glyphs and data, along with their respective signatures, are placed into the subsetted font file. This technique is less preferred, however, because it would dramatically increase the size of the subsetted font file. Glyph data typically comprise 100 to 300 bytes each. By comparison, a digital signature might be 100 bytes long. Adding a 100-byte signature to each glyph significantly enlarges the original font file. For subsetted fonts, the preferred technique requires one signature and (typically) $\log_2$ n authentication values, where n is the number of glyphs in the original font file and the size of an authentication value is typically 20 bytes. A typical range for n is 600 (for Pan-European fonts) to 16000 (for Eastern fonts). Thus, the total number of bytes needed for a typical subset is at most $100+((\log_2 16000)*20)$, or about 380. By comparison, the less preferred technique requires 100 bytes for each present glyph. So, when there are at least four present glyphs in the subset, the space requirement of the less preferred technique is greater.

This invention has been described in the context of subsetting font files. It is noted that aspects of this invention may be extended to other types of subsetted media, including video, audio, and graphics. In the case of video, for example, the video is segmented into discrete media segments (such as one second segments) and the signing module constructs an authentication tree from the media segments. Then, when a client requests a video clip (such as a 20-second clip), the server provides a subsetted media file containing the media segments in the video clip and authentication values representing the media segments not contained within the media subset. The client reconstructs the authentication tree from this information and verifies the authenticity of the video clip.

Although the invention has been described in language specific to structural features and/or methodological steps, it is to be understood that the invention defined in the appended claims is not necessarily limited to the specific features or steps described. Rather, the specific features and steps are disclosed as preferred forms of implementing the claimed invention.

What is claimed is:

1. A method for supplying authenticated font subsets, comprising the following steps:

constructing an authentication tree for a font;

subsetting the font to form a font subset; and

distributing the font subset together with at least one authentication value of the authentication tree that represents portions of the font that are not contained in the font subset.

2. A method as recited in claim 1 wherein the constructing step comprises the following steps:

constructing one or more intermediate authentication trees for tables in a font file; and

constructing the authentication tree from the intermediate authentication trees.

3. A method as recited in claim 1 further comprising the following steps:

receiving the font subset and the authentication value(s); and

authenticating the font subset using the authentication value(s).

4. A method as recited in claim 1 further comprising the following steps:

digitally signing a root of the authentication tree;

storing the font subset, the authentication value(s), and the digitally signed root as a font subset data file.

5. A data structure stored on a computer-readable medium constructed as a result of the following steps:

constructing an authentication tree for a font;

digitally signing a root of the authentication tree;

subsetting the font to form a font subset;

storing the font subset, at least one authentication value of the authentication tree that represents portions of the font that are not contained in the font subset, and the digitally signed root as a font subset data file.

6. A computer-readable medium having computer executable instructions that, when executed on a processor, perform the following steps:

constructing an authentication tree for a font;

subsetting the font to form a font subset; and

distributing the font subset together with at least one authentication value of the authentication tree that represents portions of the font that are not contained in the font subset.

7. A method comprising the following steps:

receiving a font subset that is subsetted from a font file, the font file being represented by an authentication tree whose root is digitally signed by a source;

receiving at least one authentication value from the authentication tree that represents portions of the font that are not contained in the font subset;

receiving the digitally signed root;

reconstructing the authentication tree from the font subset and the authentication value(s); and

authenticating the font subset based upon the reconstructed authentication tree and the digitally signed root.

8. A method as recited in claim 7, wherein the authenticating step comprises the following steps:

producing an unsigned root from the signed root;

deriving a reconstructed root from the reconstructed authentication tree;

comparing the reconstructed root with the unsigned root; and

accepting the font subset as authentic if the reconstructed root matches the unsigned root.

9. A computer-readable medium having computer executable instructions for performing the steps in the method as recited in claim 7.

10. A method for supplying a font comprising the following steps:

constructing a hash tree for the font, the hash tree having leaves formed of glyphs and data that define the font, the hash tree also having a root;

digitally signing the root of the hash tree to authenticate the font as belonging to a source;

subsetting the font by selecting one or more of the glyphs and data to form a font subset;

storing the font subset, one or more hash values that represent all remaining glyphs and data that are not part

of the selected font subset, and the digitally signed root as a font subset file;

electronically transmitting the font subset file;

receiving the font subset file;

producing an unsigned root of the hash tree from the signed root contained in the font subset file;

reconstructing the hash tree from the font subset and the one or more hash values contained in the font subset file to derive a reconstructed root; and

authenticating the font subset by comparing the reconstructed root with the unsigned root derived from the signed root.

11. In a system for electronically delivering media wherein the media is segmented into multiple individual media segments and the segments are used to construct an authentication tree that is digitally signed by a source to authenticate the media as belonging to the source, a method comprising the following steps:

subsetting the media to form a media subset containing one or more of the media segments; and

sending the media subset and at least one authentication value of the authentication tree that represents the media segments not contained within the media subset so that a recipient can authenticate the media subset as belonging to the source.

12. A method as recited in claim **11**, further comprising the following steps:

receiving the media subset and the authentication value(s) at the recipient; and

authenticating the media subset using the authentication value(s).

13. A method as recited in claim **11**, further comprising the following steps:

subsetting the media subset to form a second media subset containing one or more of the media segments in the media subset; and

sending the second media subset and at least one authentication value of the authentication tree that represents the media segments not contained within the second media subset so that a recipient can authenticate the second media subset as belonging to the source.

14. In a system for electronically delivering media wherein the media is segmented into multiple individual media segments and the segments are used to construct an authentication tree that is digitally signed by a source to authenticate the media as belonging to the source, a computer-readable medium having computer executable instructions that, when executed on a processor, perform the following steps:

subsetting the media to form a media subset containing one or more of the media segments; and

sending the media subset and at least one authentication value of the authentication tree that represents the media segments not contained within the media subset so that a recipient can authenticate the media subset as belonging to the source.

15. A system for supplying authenticated font subsets, comprising:

a signing module to construct an authentication tree for a font and digitally signing a root of the authentication tree;

a subsetting module to subset the font to form a font subset and constructing a font subset file containing the font subset, the digitally signed root, and at least one authentication value of the authentication tree that represents portions of the font that are not contained in the font subset; and

an authentication module to authenticate the font subset by reconstructing the root of the authentication tree using the font subset and the authentication value(s) from the font subset file, producing an unsigned root from the digitally signed root of the font subset file, and checking the unsigned root against the reconstructed root.

16. A system as recited in claim **15**, wherein the signing module resides at a font creator, the subsetting module resides at a font distributor separate from the font creator, and the authentication module resides at a recipient.

17. A system as recited in claim **15**, wherein the signing module and the subsetting module reside at a single source and the authentication module resides at a recipient.

18. A system as recited in claim **15**, further comprising:

a source computer residing at a font creator, the signing module being configured as a software component executing on the source computer;

a distributor computer residing at a font distributor, the subsetting module being configured as a software component executing on the distributor computer; and

a recipient computer residing at a recipient, the authentication module being configured as a software component executing on the recipient computer.

19. A computer program embodied on a computer-readable medium for creating a font file, the font file including a set of glyphs, comprising:

a code segment to instruct a computer to construct an authentication tree having leaves formed of glyphs, one or more intermediate levels of nodes computed as one-way functions of the glyphs, and a root computed as a one-way function of the nodes; and

a code segment to instruct a computer to digitally sign the root of the authentication tree.

20. A computer program as cited in claim **19**, wherein the font file also contains other data in addition to the glyphs, further comprising a code segment to instruct a computer to construct an authentication tree having leaves formed of glyphs and the other data, one or more intermediate levels of nodes computed as one-way functions of the glyphs and the other data, and a root computed as a one-way function of the nodes.

21. A computer operating system embodied on a computer-readable medium having code segments for creating a font file, the font file including a set of glyphs, the computer operating system comprising:

a code segment to instruct a computer to construct an authentication tree having leaves formed of glyphs, one or more intermediate levels of nodes computed as one-way functions of the glyphs, and a root computed as a one-way function of the nodes; and

a code segment to instruct a computer to digitally sign the root of the authentication tree.

22. A computer program embodied on a computer-readable medium for subsetting a font file, the font file including a set of glyphs and a digitally signed root of an authentication tree that is derived from the glyphs, comprising:

a code segment to instruct a computer to create a font subset file that includes part of the glyphs in the font file and excludes remaining ones of the glyphs in the font file; and

a code segment to instruct a computer to add to the font subset file the digitally signed root and one or more

nodes from the authentication tree that represent the remaining glyphs.

23. A computer operating system embodied on a computer-readable medium having code segments for subsetting a font file, the font file including a set of glyphs and a digitally signed root of an authentication tree that is derived from the glyphs, the computer operating system comprising:

a code segment to instruct a computer to create a font subset file that includes part of the glyphs in the font file and excludes remaining ones of the glyphs in the font file; and

a code segment to instruct a computer to add to the font subset file the digitally signed root and one or more nodes from the authentication tree that represent the remaining glyphs.

24. A computer program embodied on a computer-readable medium for handling a font subset file that is subsetted from a font file, wherein the font file is represented by an authentication tree having leaves formed of glyphs, one or more intermediate levels of nodes computed as one-way functions of the glyphs, and a root computed as a one-way function of the nodes, and wherein the font subset file contains selected glyphs, the root of the authentication tree, and one or more of the nodes from the authentication tree that represent non-selected glyphs, comprising:

a code segment to instruct a computer to reconstruct the root of the authentication tree from the selected glyphs and the nodes contained in the font subset file; and

a code segment to instruct a computer to authenticate the font subset file based on the reconstructed root and the root contained in the font subset file.

25. A computer program as recited in claim 24, wherein the root is digitally signed and the font subset file contains the digitally signed root, comprising:

a code segment to instruct a computer to produce an unsigned root from the digitally signed root contained in the font subset file;

a code segment to instruct a computer to compare the unsigned root from the font subset file to the reconstructed root; and

a code segment to instruct a computer to authenticate the font subset file if the unsigned root matches the reconstructed root.

26. A computer operating system embodied on a computer-readable medium incorporating the computer program as recited in claim 24.

27. A data structure embodied on a computer-readable medium, comprising:

an identification field containing an identification of a font subset that is subsetted from a font;

a content field containing selected glyphs and data that define the font subset;

an authentication field containing one or more authentication values from an authentication tree for the font, the authentication values representing non-selected glyphs and data from the font that are not part of the font subset; and

a signature field containing a digitally signed root value computed from a root of the authentication tree.

* * * * *

US005859648A

United States Patent [19]

Moore et al.

[11] Patent Number: 5,859,648

[45] Date of Patent: Jan. 12, 1999

[54] **METHOD AND SYSTEM FOR PROVIDING SUBSTITUTE COMPUTER FONTS**

[75] Inventors: **George M. Moore**, Redmond; **Dennis Richard Adler**; **Eliyezer Kohen**, both of Mercer Island, all of Wash.

[73] Assignee: **Microsoft Corporation**, Redmond, Wash.

[21] Appl. No.: **897,374**

[22] Filed: **Jul. 21, 1997**

Related U.S. Application Data

[63] Continuation of Ser. No. 527,291, Sep. 12, 1995, abandoned, which is a continuation of Ser. No. 85,482, Jun. 30, 1993, abandoned.

[51] **Int. Cl.**[6] **G06F 17/21**

[52] **U.S. Cl.** **345/471**; 707/542; 707/519

[58] **Field of Search** 345/467–469, 345/471, 472; 707/519, 542

[56] **References Cited**

U.S. PATENT DOCUMENTS

3,964,591	6/1976	Hill et al.	395/110
4,591,999	5/1986	Logan	707/519
4,675,830	6/1987	Hawkins	345/469 X
4,933,866	6/1990	Markoff et al.	345/471
4,987,550	1/1991	Leonard et al.	345/471
5,042,075	8/1991	Sato	345/471 X
5,099,435	3/1992	Collins et al.	345/428
5,167,013	11/1992	Hube et al.	395/110
5,185,818	2/1993	Warnock	707/542 X
5,218,460	6/1993	Nakajima	358/456
5,257,016	10/1993	Fuji et al.	345/471
5,274,365	12/1993	Martinez et al.	345/471 X
5,281,959	1/1994	Martinez et al.	345/471 X
5,291,186	3/1994	Martinez et al.	345/471 X
5,304,989	4/1994	Martinez et al.	345/471
5,319,358	6/1994	Martinez et al.	345/471 X
5,325,479	6/1994	Kaasila	345/469
5,495,577	2/1996	Davis et al.	707/542
5,506,940	4/1996	Bamford et al.	395/110
5,533,174	7/1996	Flowers, Jr. et al.	395/114
5,548,695	8/1996	Asano et al.	345/433
5,664,086	9/1997	Brock et al.	345/468

FOREIGN PATENT DOCUMENTS

0478339 A1	4/1992	European Pat. Off.	G06F 17/21
0518554 A2	12/1992	European Pat. Off.	G06F 17/21

OTHER PUBLICATIONS

Seybold, Microsoft Licenses Panose Font Scheme, Seybold Report on Desktop Publishing, Jun. 17, 1991, pp. 40–41.

"Adobe's Super ATM:Super Font Swapper," *MacUser*, Feb. 1993, p. 47.

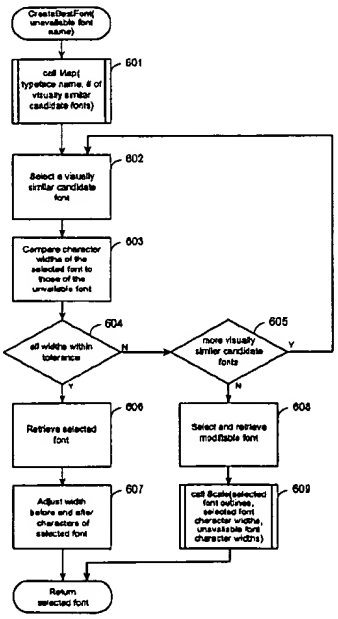
Primary Examiner—Anton Fetting

Attorney, Agent, or Firm—Seed and Berry LLP

[57] **ABSTRACT**

A method and system for providing a substitute font that visually approximates a selected font that is unavailable in a computer system is provided. In a preferred embodiment of the present invention, the method and system first selects as the substitute font a font that is available in the computer system. The method and system then adjusts the overall widths of the characters of the substitute font to match the overall widths of the corresponding characters of the selected font. This causes the same combinations of characters of the substitute font and of the selected font to have substantially the same size and appearance. The method and system then makes the substitute font available to a program that has requested the selected font.

61 Claims, 9 Drawing Sheets



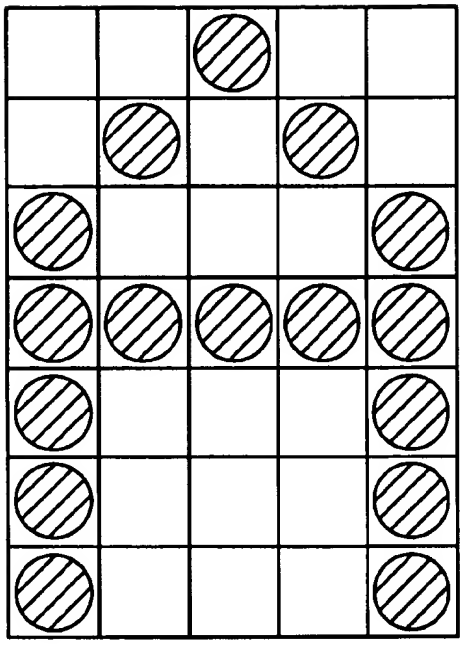
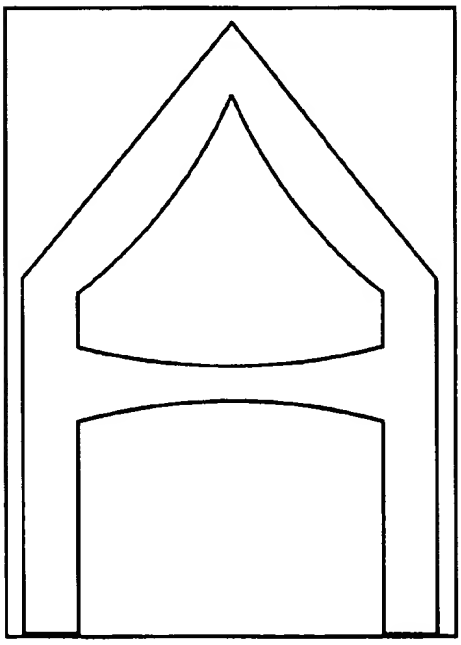


Fig. 1A



Prior Art

Fig. 1B

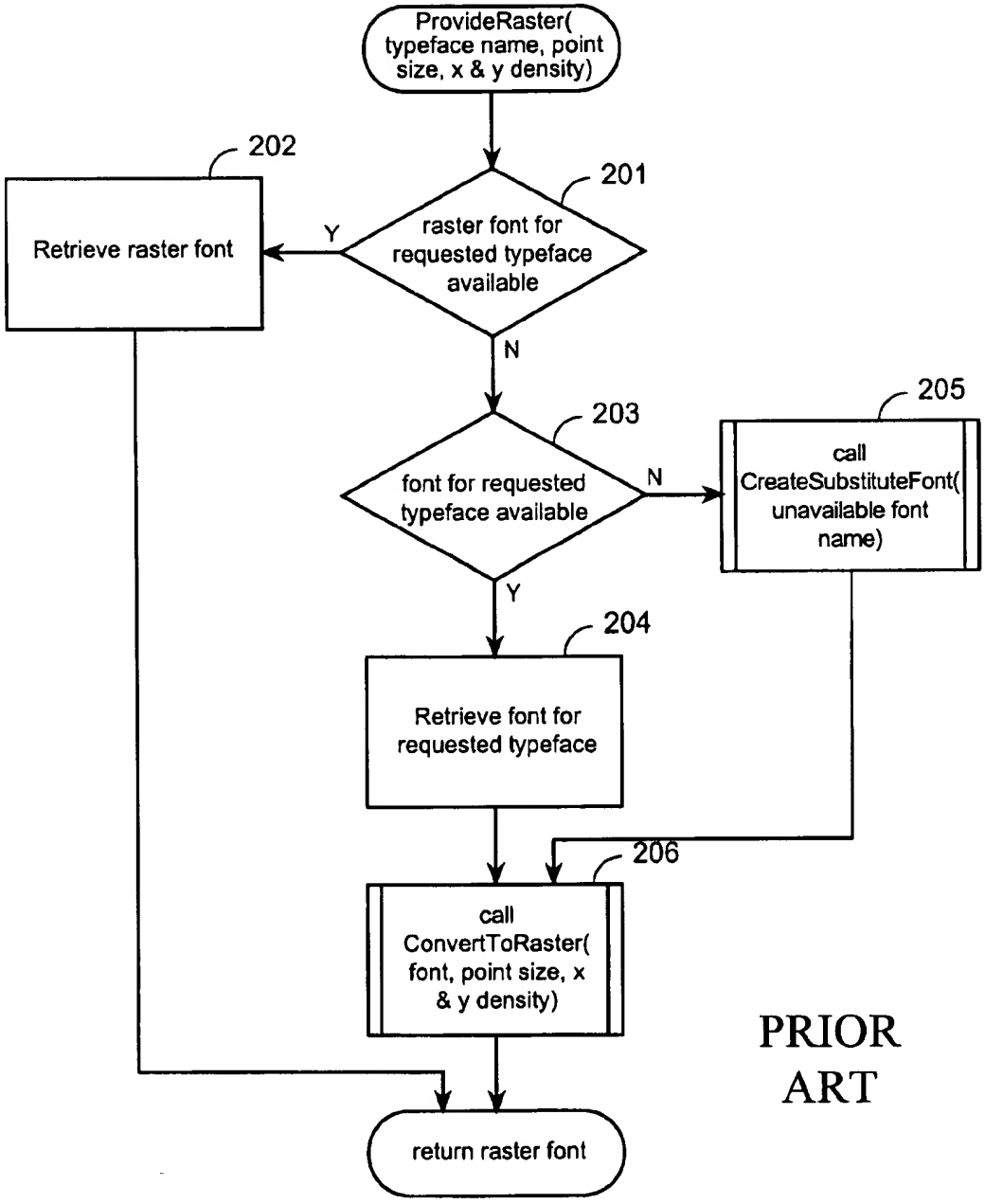
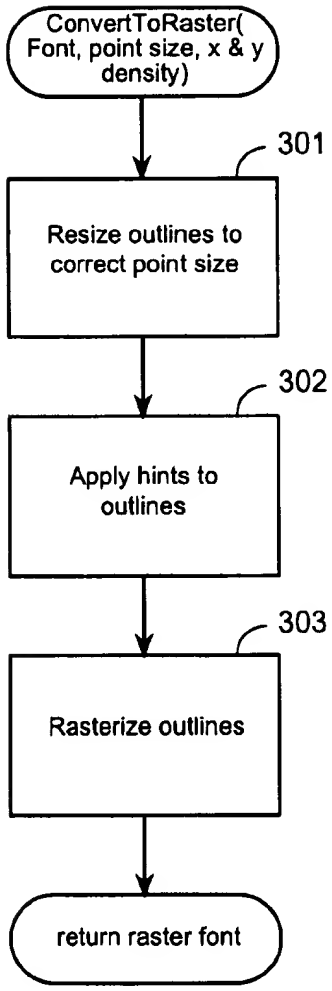
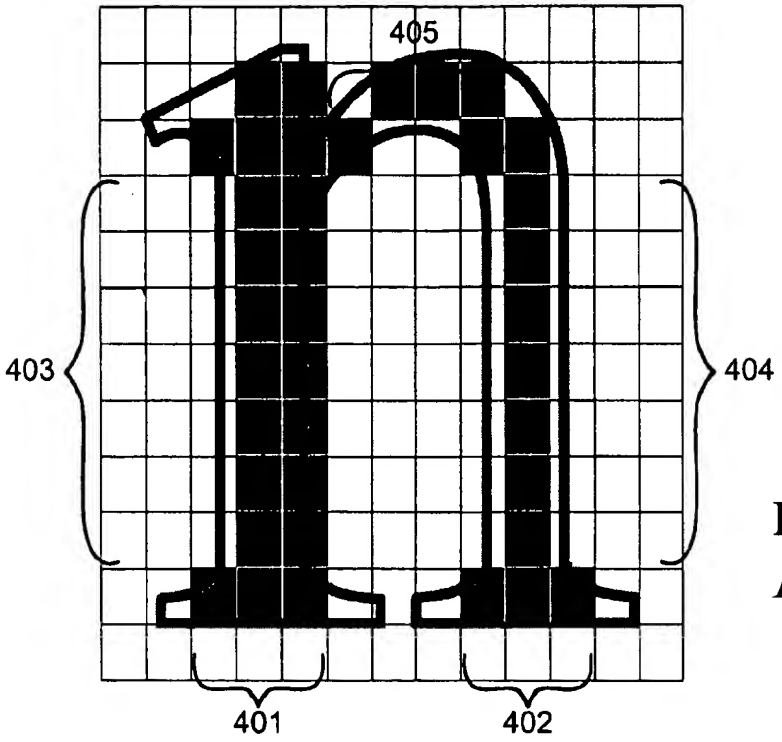

ProvideRaster(typeface name, point size, x & y density)
201
raster font for requested typeface available
Y
202
Retrieve raster font
N
203
font for requested typeface available
N
205
call CreateSubstituteFont(unavailable font name)
Y
204
Retrieve font for requested typeface
206
call ConvertToRaster(font, point size, x & y density)
return raster font
PRIOR ART

Fig. 2



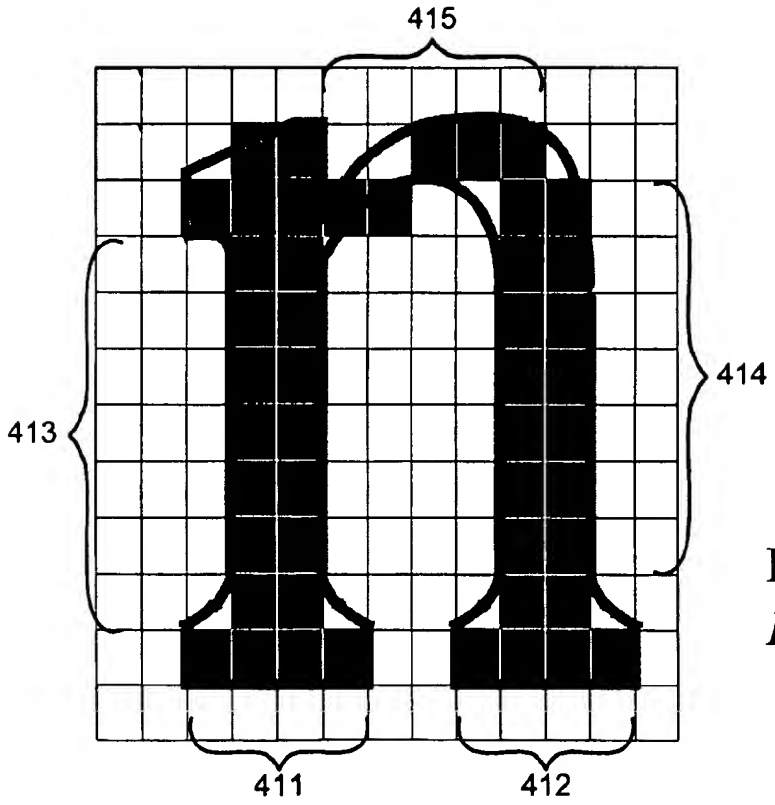
PRIOR ART

Fig. 3



Prior Art

Fig. 4A



Prior Art

Fig. 4B

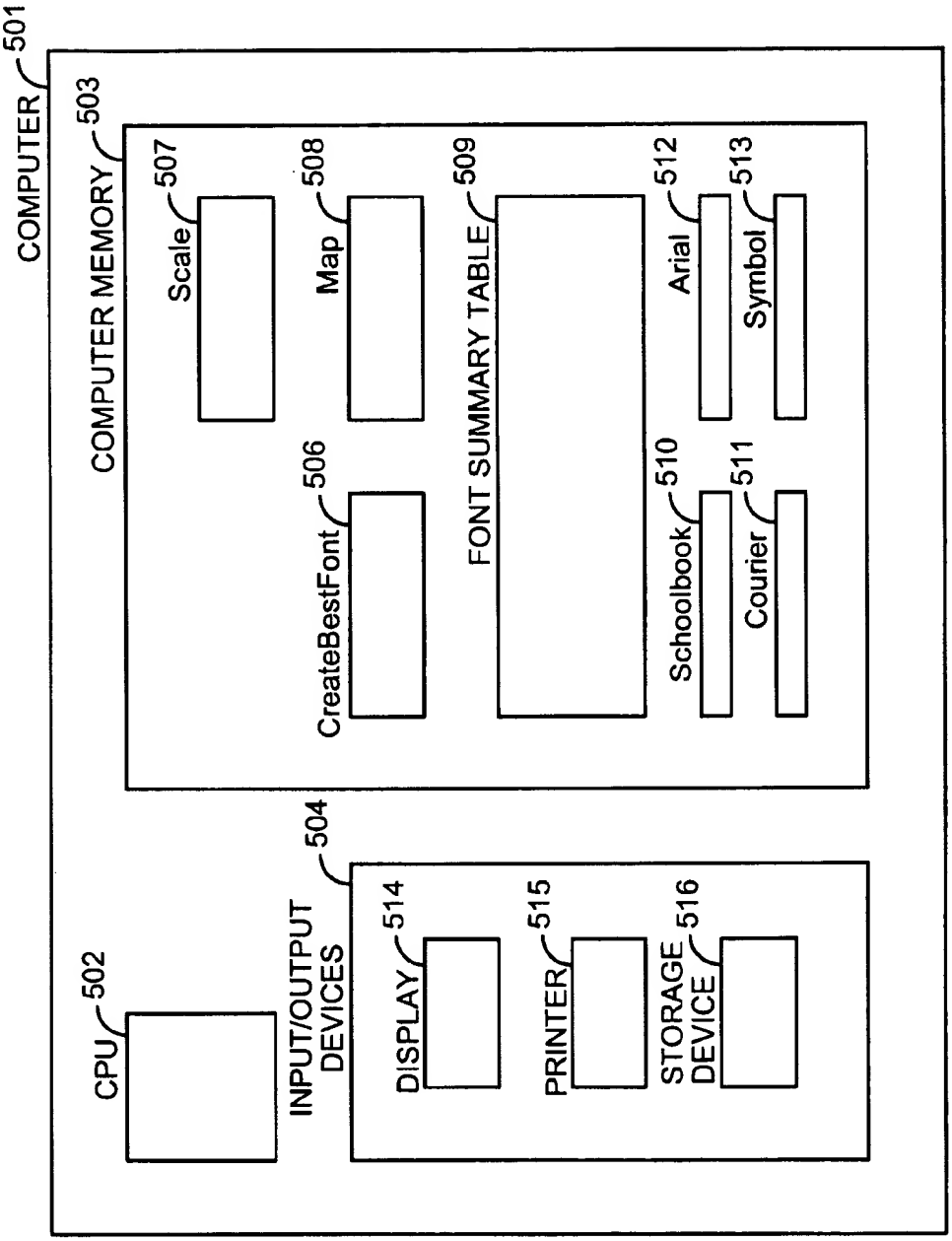

COMPUTER
501
COMPUTER MEMORY
503
Scale
507
Map
508
FONT SUMMARY TABLE
509
Arial
512
Symbol
513
CreateBestFont
506
Schoolbook
510
Courier
511
INPUT/OUTPUT DEVICES
504
DISPLAY
514
PRINTER
515
STORAGE DEVICE
516
CPU
502

Fig. 5

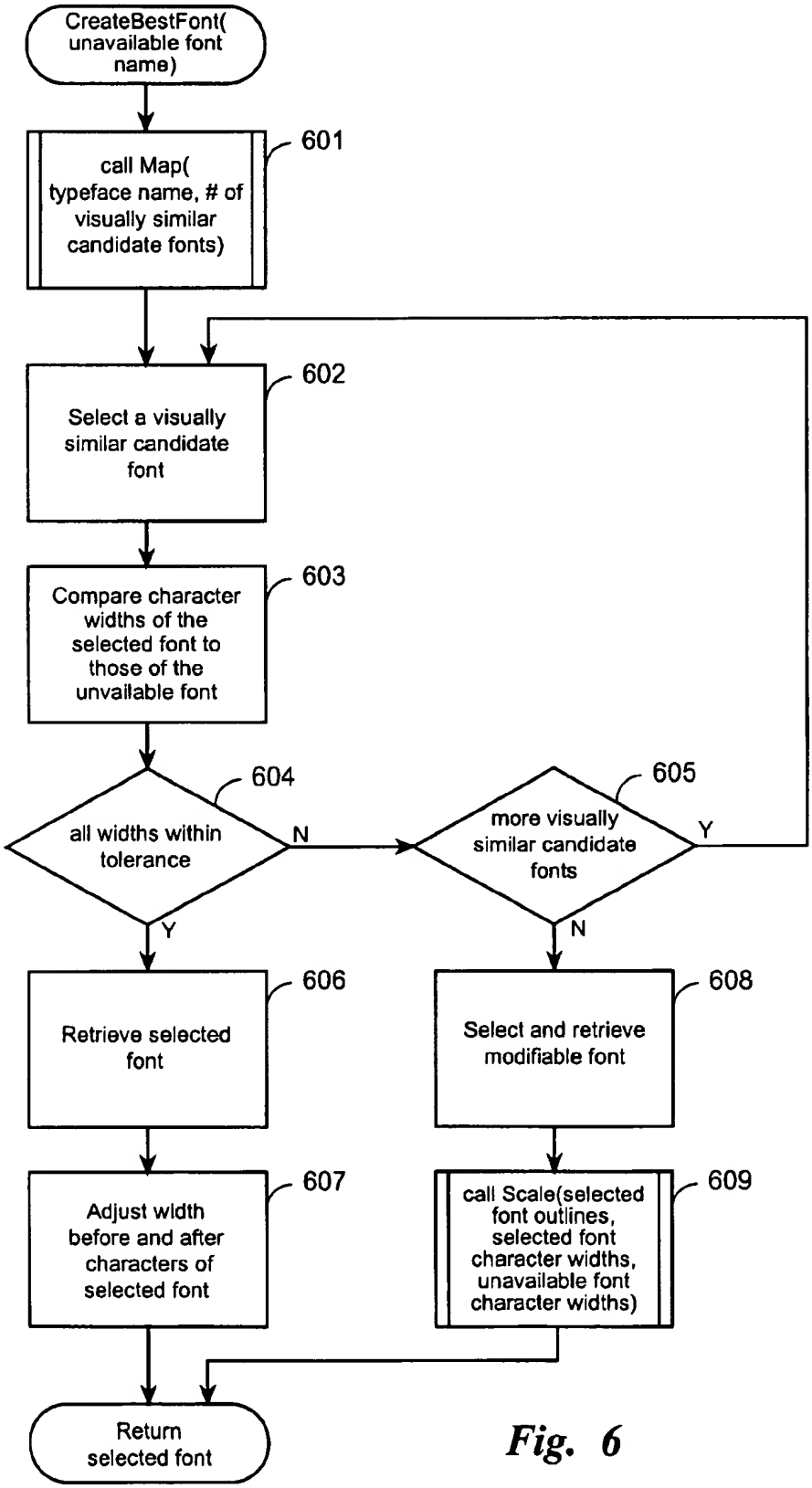
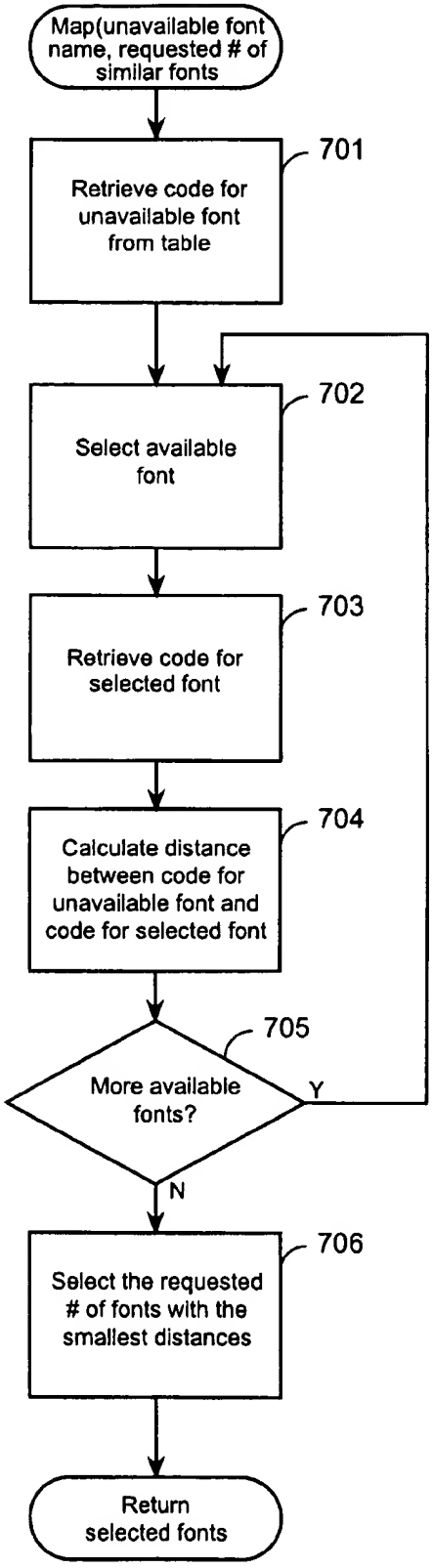

CreateBestFont(unavailable font name)
call Map(typeface name, # of visually similar candidate fonts)
601
Select a visually similar candidate font
602
Compare character widths of the selected font to those of the unvailable font
603
all widths within tolerance
604
N
Y
more visually similar candidate fonts
605
Y
N
Retrieve selected font
606
Select and retrieve modifiable font
608
Adjust width before and after characters of selected font
607
call Scale(selected font outlines, selected font character widths, unavailable font character widths)
609
Return selected font

Fig. 6



PRIOR ART

Fig. 7

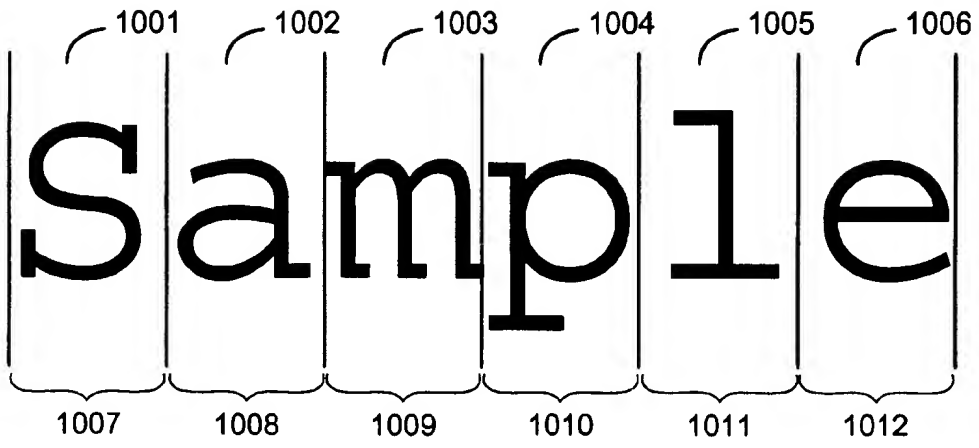

1001
1002
1003
1004
1005
1006
Sample
1007
1008
1009
1010
1011
1012

Fig. 10

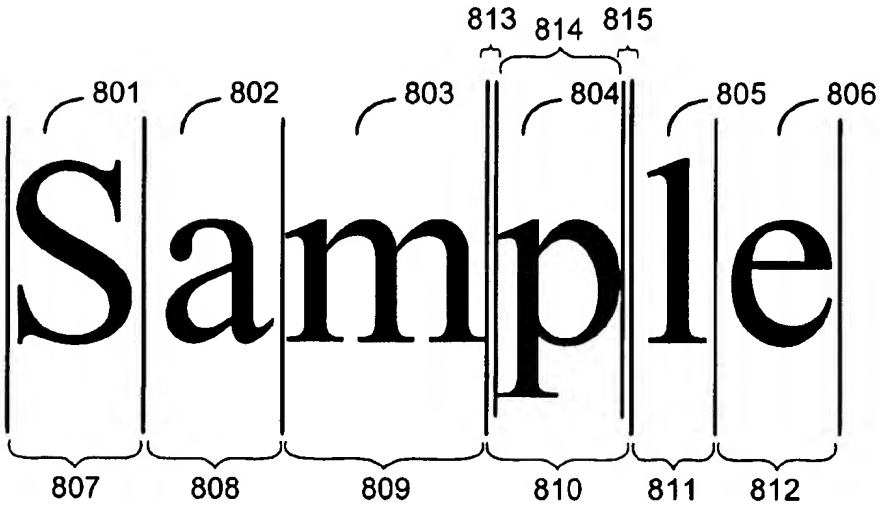

813
814
815
801
802
803
804
805
806
Sample
807
808
809
810
811
812

Fig. 8

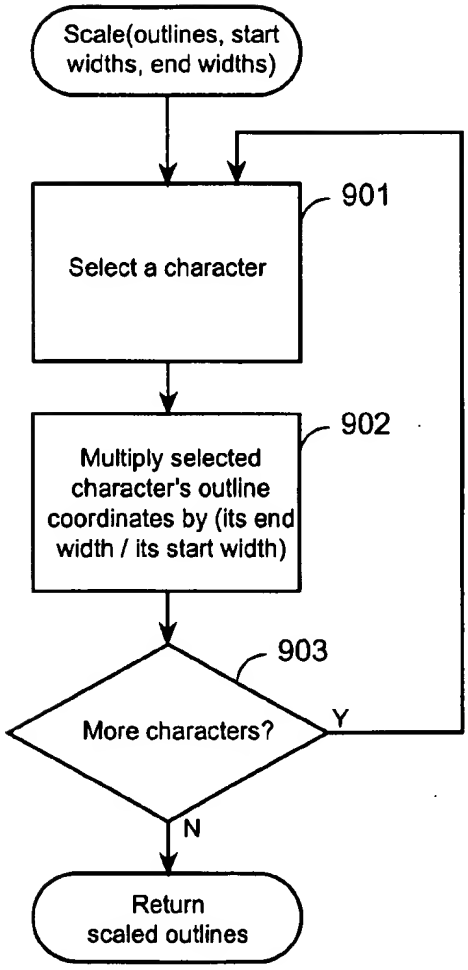

Scale(outlines, start widths, end widths)
901
Select a character
902
Multiply selected character's outline coordinates by (its end width / its start width)
903
More characters?
Y
N
Return scaled outlines

Fig. 9

METHOD AND SYSTEM FOR PROVIDING SUBSTITUTE COMPUTER FONTS

CROSS-REFERENCE TO RELATED APPLICATIONS

This application is a continuation of U.S. patent application Ser. No. 08/527,291, filed Sep. 12, 1995, for "METHOD AND SYSTEM FOR PROVIDING SUBSTITUTE COMPUTER FONTS", now abandoned, which is a continuation of application Ser. No. 08/085,482, filed Jun. 30, 1993, for "METHOD AND SYSTEM FOR PROVIDING SUBSTITUTE COMPUTER FONTS (CREATEBESTFONT),"now abandoned.

TECHNICAL FIELD

The invention relates generally to a method and system for providing computer fonts, and, more specifically, to a method and system for providing a substitute computer font for a font that is unavailable within a computer system.

BACKGROUND OF THE INVENTION

Most general purpose computer systems (computers) are able to display text, comprised of characters (letters, numerals, and other symbols), on attached display devices, such as monitors and printers. A monitor consists of a grid of points, or pixels, each of which can be individually illuminated. Computer printers and other computer display devices also use pixels to display images.

In order to display text on a display device, computers use one or more raster fonts. A raster font is comprised of character bitmaps. Each character bitmap contains data reflecting which pixels in a subset of the display grid must be illuminated in order to form a particular character. When a computer needs to display a particular character at a display location, it accesses the bitmap for that character. The computer then turns the illumination of pixels near the display position on or off in accordance with the data stored in the character bitmap. FIG. 1A is a diagram showing how one possible bitmap of the letter A could be displayed on a display device.

At one point, computers each used only a single raster font, permanently stored in read only memory. Since then, computers have been adapted to use several different raster fonts simultaneously. This permits a user to display text in more than one typeface and point size. (A typeface is a specific design for a set of characters. Point size is a measurement of the height of a font's characters.) These raster fonts are now stored in writeable random access memory, which allows existing raster fonts to be altered and new raster fonts added.

Computers now also use another, more robust type of font, called outline fonts. Outline fonts contain an outline for each character, comprised of straight lines and curves that form the shape of the character's outline. FIG. 1B is a diagram showing one possible outline of the letter A. The outline font describes these straight lines and curves in terms of an arbitrary system of coordinates. The computer can resize these character outlines to any point size, then convert them to a raster font. Outline fonts also contain hints, which are routines that, when executed, adjust the shapes of the outlines for various point sizes to improve their appearance.

Computers use outline fonts to create raster fonts in various sizes. Computers also transmit outline fonts to smart display devices that are themselves able to create raster fonts from the outline fonts. Many computers now store several outline fonts, each corresponding to a different typeface.

Using outline fonts, computers can now create raster fonts dynamically in different typefaces in response to the needs of users. Computer programs like ProvideRaster provide this service. FIG. 2 is a flow diagram of the ProvideRaster program. This program is an example of a font manager program that is called by application programs (applications), such as a word processor or a spreadsheet, when they require a raster font. The program is passed a requested typeface name, point size, and horizontal and vertical densities of the intended display device. The program returns a raster font of the requested typeface and point size for the intended display device.

In step **201**, if a raster font of the correct point size and horizontal and vertical pixel density is available within the computer, then the program continues at step **202**, else the program continues at step **203**. In step **202**, the program retrieves the appropriate raster font. The program then returns the retrieved raster font to the application program.

In step **203**, if an outline font for the requested typeface is available within the computer, then the program continues in step **204**, else the program continues in step **205** to create a substitute font for the unavailable outline font. In step **204**, the program retrieves the outline font for the requested typeface, which contains both character outlines and hints. The program then continues at step **206**.

In step **205**, the program calls CreateSubstituteFont to produce a substitute outline font that visually approximates the unavailable outline font. The program passes CreateSubstituteFont the unavailable outline font name. The CreateSubstituteFont program returns a substitute outline font for the unavailable font, which contains both character outlines and hints. The details of CreateSubstituteFont are discussed further below. When CreateSubstituteFont returns, the program continues at step **206**.

In step **206**, the program calls a ConvertToRaster program, passing it the outline font retrieved or created, the requested point size, and the requested horizontal and vertical densities. The program then returns the raster font created by ConvertToRaster to the application program.

FIG. **3** is a flow diagram of ConvertToRaster. The program receives as parameters an outline font, a requested point size, and a requested horizontal and vertical density. The outline font contains character outlines and hints.

In step **301**, the program resizes the character outlines of the outline font to the requested point size by multiplying the coordinates that define the outlines' lines and curves by the requested point size. This alters the height and width of each character. In step **302**, the program applies the hints of the outline font to the magnified outlines.

FIGS. **4A–B** are screen images demonstrating the need to use hints to improve the appearance of rasterized outlines. FIG. **4A** is a screen image showing a letter "n" that has been rasterized without first being hinted. The unhinted letter's left base serif **401** does not match its right base serif **402**. Further, its left vertical stem **403** is wider than its right vertical stem **404**. Finally, its crown **405** is skewed toward the top row of pixels. FIG. **4B** is a screen image showing a letter "n" that was rasterized after being hinted. The hinted letter's base serifs **411** and **412** are of the same width relative to their respective vertical stems **413** and **414**. Those vertical stems are both now two pixels wide. The height of its crown **415** has been balanced by moving the vertical stems apart slightly. The hints demonstrated by FIG. **4B** ensure the symmetry and regularity of the characters of a font. Other traditional hints ensure the vertical alignment of corresponding features of characters. Further traditional hints ensure

good color—that is, consistent weighting among characters of a font. Applying hints to a character is said to regularize the character.

In step **303**, the program rasterizes the hinted outlines of the outline font by superimposing a grid corresponding to the horizontal and vertical density of the intended display device over the hinted outlines, then turning on any pixels of the grid whose centers fall within the hinted outlines. This technique is well known in the art of digital typography. The program returns the resulting raster font to the application program, which may use it to display text. The raster font may also be stored in a buffer to be used to satisfy any identical requests received in the future.

The function served by CreateSubstituteFont is quite important. It is difficult for a computer user to anticipate which outline fonts he or she will need. Outline fonts will hereafter be referred to simply as fonts. Since individual fonts can be expensive to license, and even more expensive to independently develop, it has become quite useful for a computer to be able to construct a substitute font for an unavailable font.

The substitute fonts created by CreateSubstituteFont should: (1) be similar in appearance to the unavailable font, if possible; and (2) have exactly the same overall character widths as the unavailable font. These requirements are dictated by the paradigm of using the substitute font in place of the unavailable font to display a particular section of text in a document. The first requirement ensures that a sense of any aesthetics intended for the section will be conveyed. The second requirement ensures that lines and pages will break at the correct points.

Existing implementations of CreateSubstituteFont work by selecting an available font (the basis font) on which to base the substitute font, then modifying the basis font to create the substitute font. Basis font selection processes vary, as do their effectiveness at selecting a basis font with an appearance similar to the unavailable font.

Modification processes typically consist only of adding space before or after each character of the basis font whose overall width is smaller than the corresponding character of the unavailable font. Such implementations produce fonts with the same overall character widths in the case of characters of the basis font whose overall widths are smaller than or equal to those of the corresponding characters of the unavailable font. Such implementations cannot, however, produce fonts with the same overall character widths in the case of characters of the basis font whose overall widths are larger than those of the corresponding character of the unavailable font. It can be seen that these implementations of CreateSubstituteFont satisfy the first requirement for substitute fonts to varying degrees, and the second, more important requirement only sporadically.

SUMMARY OF THE INVENTION

It is an object of the present invention to provide a method and system for adjusting the overall widths of characters of a substitute font to match the overall widths of corresponding characters of an unavailable font.

It is another object of the present invention to provide a method and system for adjusting the overall widths of the characters of a substitute font by scaling each character horizontally.

It is a further object of the present invention to provide a method and system for adjusting the overall widths of the characters of a substitute font by adjusting the leading width before and the leading width after each character.

These and further objects, which will become apparent as the invention is more fully described below, are obtained by an improved method and system for providing a substitute font that visually approximates a selected font that is unavailable in a computer system. In a preferred embodiment of the present invention, the method and system first selects a font that is available in the computer system as the substitute font. The method and system then adjusts the overall widths of the characters of the substitute font to match the overall Widths of the corresponding characters of the selected Font. This causes the same combinations of characters of the substitute font and of the selected font to have substantially the same size and appearance. The method and system then makes the substitute font available to a program that has requested the selected font.

BRIEF DESCRIPTION OF THE DRAWINGS

FIG. **1A** is a diagram showing how one possible bitmap of the letter A could be displayed on a display device.

FIG. **1B** is a diagram showing one possible outline of the letter A.

FIG. **2** is a flow diagram of ProvideRaster.

FIG. **3** is a flow diagram of ConvertToRaster.

FIG. **4A** is a screen image showing a letter "n" that has been rasterized without first being hinted.

FIG. **4B** is a screen image showing a letter "n" that was rasterized after being hinted.

FIG. **5** is a high-level diagram of the general-purpose computer system on which the CreateBestFont, Map, and Scale programs preferably execute.

FIG. **6** is a flow diagram of CreateBestFont.

FIG. **7** is a flow diagram of a well-known embodiment of Map.

FIG. **8** is a diagram showing the overall width of a sample character, and its relationship to other horizontal measures of the character.

FIG. **9** is a flow diagram of Scale.

FIG. **10** is a diagram showing the overall width of sample characters of a font called Courier.

DETAILED DESCRIPTION OF THE INVENTION

The present invention relates to a method and system for providing a substitute outline font for an outline font that is unavailable within a computer system. outline fonts will hereafter be referred to simply as fonts. In a preferred embodiment of the present invention, the ProvideRaster program calls a CreateBestFont program (CreateBestFont) in place of CreateSubstituteFont to produce a substitute font for an unavailable font.

CreateBestFont is passed the unavailable font name. The program proceeds by attempting to identify a visually similar basis font selected by a Map program whose character widths are all within a predetermined tolerance of the unavailable font's character widths. If such a basis font exists, the program adjusts the space before and after every character of the basis font so that the overall width of each of its characters is equal to that of the corresponding character of the unavailable font. The resulting substitute font has an appearance that is similar to that of the unavailable font; the same overall character widths as the unavailable font; and the same color as the basis font, which is presumably good.

Otherwise, the program selects a font whose characters it can modify to be the correct width. It then calls a Scale

program (Scale) to do the actual modification. The resulting substitute font has an appearance that is somewhat similar to that of the unavailable font, the same overall character widths as the unavailable font, and good color. Modifiable fonts contain special hints that ensure that the modifiable fonts will have consistent stem width when modified.

FIG. **5** is a high-level diagram of the general purpose computer system on which the CreateBestFont, Map, and Scale programs preferably execute. The computer **501** contains a central processing unit (cpu) **502**, a computer memory **503**, and input/output devices **504**. The computer memory contains the CreateBestFont program **506**, the Scale program **507**, the Map program **508**, a font summary table (table) **509**, and fonts **510–513**. The input/output devices include a display device **514**, a printer **515**, and a storage device **516** such as a hard disk drive. The font summary table contains summary information on a large number of popular fonts. Each row of the table corresponds to a single font, and contains that font's name, the overall width of each of the font's characters, an indication of whether the font is a serif font or a sans serif font, and an encoded visual appearance assessment for the font.

Programs such as ProvideRaster call CreateBestFont to produce a substitute font for an unavailable font, passing it the unavailable font name. CreateBestFont proceeds by attempting to identify a visually similar font whose character widths are all within a predetermined tolerance of the unavailable font's character widths. If such a basis font exists, the program adjusts the space before and after every character of the basis font so that the width of each of its characters is equal to that of the corresponding character of the unavailable font. Otherwise, the program selects a modifiable font whose characters it can modify to be the correct width. It then calls Scale to do the actual modification. Modifiable fonts contain special hints that ensure that the resultant fonts will have good color.

FIG. **6** is a flow diagram of CreateBestFont. In step **601**, the program calls the Map program to identify visually similar fonts that are available within the computer. The program passes Map the unavailable font name and a predetermined number of candidate fonts to return. Map uses the encoded visual appearance assessments (codes) for the most popular fonts in general use stored in the font summary table. Fonts and their codes may be included in the table even if no font is available for them within the computer. In a preferred embodiment, the codes are determined in accordance with the ElseWare PANOSE™ font matching system. Codes determined in accordance with the PANOSE system contain information on family kind, serif style, weight, proportion, contrast, stroke variation, arm style, letter form, midline, and X-height. In a further preferred embodiment, codes for at least 100 fonts are included in the table.

FIG. **7** is a flow diagram of a well-known embodiment of Map. In step **701**, the program retrieves the code for the unavailable font from the font summary table. In steps **702–705**, Map calculates the distance from the unavailable font for each of the available fonts. In step **702**, the program selects an available font. In step **703**, the program retrieves the code for the selected font from the table. In step **704**, the program calculates the distance between the code for the unavailable font and the code for the selected font. This calculation involves, for each digit of the code, squaring the difference between the unavailable font digit value and the selected font digit value. The squares are then added to yield the distance. In step **705**, if more available fonts remain, then the program continues at step **702**, else the program continues at step **706**. In step **706** the program selects the requested number of fonts with the smallest distances from the unavailable font. The program then returns the selected fonts to CreateBestFont.

Table 1 below is a partial simplified view of the font summary table. It shows the font name and associated code for each of several fonts. The description below details how Map would behave if this font summary table was present on the system and Map was called with the parameters (unavailable font name=Times New Roman, requested number of similar fonts=2).

TABLE 1

Font Name	Code
Arial	2 2 2 2 2 2 2 2 2 2
Courier	0 2 2 3 4 5 7 6 8 9
Schoolbook	2 2 2 2 5 5 5 8 8 8
Symbol	9 8 7 6 5 4 3 2 1 0
Times New Roman	0 1 2 3 4 5 6 7 8 9
0	0
0	0
0	0

Map first retrieves the code for the unavailable font, Times New Roman, from the table—0123456789. Map then selects the first available font, Arial, and retrieves its code—2222222222. Then Map calculates the distance between the code for Times New Roman and the code for Arial. This calculation involves, for each code digit, squaring the difference between the unavailable font code digit and the selected font code digit. The squares are then added to yield the distance. The distance between Arial and Times New Roman is:

$$(2-0)^2+(2-1)^2+(2-2)^2+(2-3)^2+(2-4)^2+$$
$$(2-5)^2+(2-6)^2+(2-7)^2+(2-8)^2+$$
$$(2-9)^2=4+1+0+1+4+9+16+25+36+49=145.$$

This process of calculating the distance from Times New Roman is then repeated for each of any other remaining available fonts in the table. Map then selects the two fonts with the smallest distances from Times New Roman. These are Courier, with a distance of 3, and Schoolbook, with a distance of 10. Map then returns these two fonts.

In an alternate embodiment, each font available on the system is stored with a list of fonts to which it is visually similar. Each font is listed with a relative similarity rating value. In this embodiment, Map identifies fonts that are visually similar to the unavailable font by reading the list associated with each available font to identify the fonts that are similar to the unavailable font, and returning the requested number of fonts whose similarity rating values are the highest.

In steps **602–605**, the program selects each visually similar candidate font in turn, checking to see whether its character widths are within a predetermined tolerance of those of the requested font. In step **602**, the program selects a visually similar font from among the candidates returned by Map. In step **603**, the program compares the overall width of each character of the selected font to that of the corresponding character of the unavailable font.

FIG. **8** is a diagram showing the overall width of a sample character, and its relationship to other horizontal measures of the character. Characters **801–806** have been displayed by an application program displaying the word "Sample" in the Times New Roman font. The character "p" **804** has an

overall width **810**, made up of a leading width **813**, a horizontal extent **814**, and a trailing width **815**. The other characters shown also have these measures.

The overall widths of the characters of popular fonts are stored in the font summary table. In step **603**, the program accesses the overall widths from the rows of the table corresponding to the unavailable font and the selected font in order to carry out its comparison of them.

In step **604**, if all of the overall widths of the characters of the selected font are within a threshold proportion of the overall widths of the corresponding characters of the unavailable font, then the program continues at step **606** to use the selected font, else the program continues at step **605**. In step **605**, if more visually similar candidate fonts remain, then the program continues at step **602** to select one of them, else the program continues at step **608**.

In an alternate embodiment, instead of calling Map to request a relatively large number of visually similar fonts, the program calls Map to request a smaller number of visually similar fonts. In this embodiment, the program calls Map again for more visually similar fonts if no fonts in the first group had overall character widths within a threshold proportion of the overall widths of the corresponding characters of the unavailable font.

Steps **606–607** are the path for generating the substitute font when a visually similar font has acceptable character widths. In step **606**, the program retrieves the font corresponding to the selected font from memory. The font is preferably retrieved from a file stored on the storage device, using the font name. This becomes the basis font. In step **607**, the program adjusts the leading and trailing width of each character of the basis font to make its overall width equal to the overall width of the corresponding character in the unavailable font. The program then returns the basis font, and the basis font becomes the substitute font.

Steps **608–609** are the path for generating the substitute font when no visually similar font has acceptable character widths. In step **608**, the program selects and retrieves from memory a font that has been specially adapted to support the horizontal expansion and compression of the overall widths of its characters. The font is preferably retrieved from a file stored on the storage device, using the font name. These modifiable fonts have been adapted by adding special hints that ensure that the color of the font is good after expansion and/or compression by regularizing vertical stem width among characters. Special hints are discussed further in conjunction with Scale. The selected modifiable font becomes the basis font.

In a preferred embodiment, it is sufficient for there to be only two modifiable fonts with specials hints, one a font for a serif font and one a font for a sans serif font. (Characters of serif fonts have short lines or ornaments at the ends of their stems, while characters of sans serif fonts do not.) In this embodiment, the selection process simply selects the serif font as the basis font for serif unavailable fonts and the sans serif font as the basis font for sans serif unavailable fonts. An indication of whether each popular font is a serif font or a sans serif font is stored in the table. In this embodiment, the program accesses the row of the table corresponding to the unavailable font to determine whether the unavailable font is a serif or a sans serif font.

In step **609**, the program calls Scale to horizontally scale each character of the selected basis font to the correct width. The program passes the outlines of the basis font, the overall character widths of the basis font, and the overall character widths of the unavailable font to Scale. When Scale returns, the program returns the scaled basis font. This becomes the substitute font.

FIG. **9** is a flow diagram of Scale. The program receives as parameters a set of character outlines, starting overall character widths, and ending overall character widths. The program scales each character outline to the ending width specified. In steps **901–903**, the program cycles through all of the character outlines, scaling each one. In step **901**, the program selects a character outline. In step **902**, the program scales the selected character outline to the correct width by multiplying each of its outline coordinates by its starting overall character width and dividing it by its ending overall character width. In step **903**, if more character outlines remain, then the program continues at step **901** to select one of them, else the program returns the scaled outlines.

FIG. **10** is a diagram showing the overall widths of sample characters of the Courier font. The word "Sample" is formed by characters **1001–1006**, having respective overall widths **1007–1012**. Overall widths **807–812** (Times New Roman) and **1007–1012** (Courier) for the characters displayed are expressed numerically in columns 2 and 3 of Table 2 below, which lists characters alphabetically, with capitals first.

TABLE 2

Character	Courier Width	Times New Roman Width	Scale Multiplier
...			
S	100	100	1
...			
a	100	85	.85
...			
e	100	80	.8
...			
l	100	55	.55
m	100	140	1.4
...			
p	100	105	1.05
...			

If Times New Roman was the unavailable font, and CreateBestFont selected Courier as the modifiable font and called Scale in order to scale the character widths of Courier to those of Times New Roman, Scale would proceed as follows. Scale's outlines parameter contains the character outlines from the Courier font. The start widths parameter contains the overall widths for the characters of the Courier font. The end widths parameter contains the overall widths of the Times New Roman font. For each character, Scale multiplies that character's outline coordinates by the ratio of its end width over its start width. In the case of the character "S", Scale would multiply the outline coordinates by 100/100=1. The multipliers for other selected characters are shown in the fourth column of Table 2. After scaling all the outlines, Scale would return them to CreateBestFont.

The special hints included in the adapted fonts must correct irregularities caused by the horizontal scaling process. The special hints, along with the traditional hints of the basis modifiable font, are applied to the scaled outlines of the substitute font by ConvertToRaster in step **302**. In a preferred embodiment, the special hints identify the character of the selected font that was horizontally scaled the least by Scale. The special hints then extract the vertical stem width of the least scaled character and apply it to every other character in the selected font. This involves identifying the vertical stems of each character, then moving the edges of the vertical stems either together or apart so that the width of each vertical stem matches the width of the least scaled character's vertical stems.

In a further preferred embodiment, Scale stores an indication of which character it horizontally scaled the least and

the scaling factor applied to that character for retrieval by the hints. ConvertToRaster preferably executes the hints for the least scaled character first. When executed, the hints for the least scaled character set a flag associated with the font specifying that, when the hints for the other characters are executed, their special hints sections will be executed. The hints for the least scaled character further access the scaling factor and set a standard vertical stem width value, represented in font units. When the hints for each of the other characters are executed, the character's special hints section is executed. The special hints section consists of instructions that override the distance, in font units, between character control points on the left and right sides of each of the character's vertical stems with the stored vertical stem width of the least scaled character. Thus overridden, when rasterized, the vertical stems of this character will be the same number of pixels wide as those of the least scaled character, resulting in a font with even color on each of the vertical stems.

While this invention has been shown and described with reference to preferred embodiments, it will be understood by those skilled in the art that various changes or modifications in form and detail may be made without departing from the spirit and scope of this invention. For example, instead of scaling the characters of a basis font horizontally to match an unavailable font, the facility could scale the characters of the basis font vertically to match the unavailable font. In this case, special hinting would regularize horizontal stem height instead of vertical stem width. Also, CreateBestFont could use methods other than those described to identify visually similar fonts. Further, the invention could be implemented within a smart display device instead of a general purpose computer. Still further, the fonts involved need not be for Roman alphabets. The same techniques could be applied to such diverse alphabets as Russian, Hebrew, Japanese, or Korean, or to nonalphabetic symbol fonts.

We claim:

1. A method in a computer system for providing a substitute font that visually approximates a requested font that is unavailable in the computer system, the requested font and the substitute font having respective sets of characters each of which have an overall width, the overall width having a leading width, an extent width, and a trailing width, the requested font and the substitute font being outline fonts, the method comprising;

receiving a request from a requesting program for a requested font;

determining that the requested font is unavailable in the computer system;

selecting as the substitute font to replace the requested font a font that is available in the computer system;

adjusting the overall widths of the characters of the substitute font to match the overall widths of corresponding characters of the requested font by adjusting the leading and trailing widths without adjusting the extent width so that the visual appearance of each character of the substitute font is not changed; and

making the substitute font with the overall widths of the characters adjusted to match the overall widths of the requested font available to the requesting program so that when the requesting program displays characters using the substitute font and without further adjustment of the overall widths of the characters, the characters are displayed with overall widths that match the overall widths as if the characters had been displayed using the requested font.

2. The method of claim 1, further including the step of adjusting predetermined features in the characters of the substitute font prior to making the substitute font available so that the predetermined features in the characters of the substitute font are consistent with each other.

3. The method of claim 2 wherein one of the predetermined features in the characters of the substitute font is the width of stems of the characters.

4. The method of claim 3 wherein the stems that are adjusted are the vertical stems of the characters of the substitute font.

5. The method of claim 3 wherein the widths of the stems of the substitute font characters are adjusted to match the width of a stem of the substitute font character that has an overall width that is closest to the overall width of the corresponding selected font character so that the widths of the stems of the substituted fonts passed the substitute font to the program that has requested the selected font are equal.

6. The method of claim 1 wherein the computer system contains a plurality of fonts, and wherein the method further includes the step of comparing the characters of each of the fonts in the computer system to the characters of the selected font, and selecting as the substitute font a font that is visually similar to the selected font.

7. The method of claim 1, wherein the receiving step includes the step of receiving an indication of the identity of a requested font and an indication of the size at which the overall font is to be rendered, and

wherein the determining step includes the step of determining that no font having the identity indicated by the received indication of identity is available in the computer system.

8. A method in a computer system for providing a substitute font for an unavailable font having characters with overall widths the unavailable font having a numerical characterization of its visual characteristics, the method comprising the steps of:

receiving a request from a requesting program for a requested font;

determining that the requested font is unavailable in the computer system;

identifying one or more candidate fonts that are available within the computer system to replace the unavailable font having numerical characterizations of their visual characteristics that differ from that of the unavailable font by less than a preselected maximum distance;

selecting a basis font from among the candidate fonts that has characters with overall widths, each of which is the sum of an extent width corresponding to the width of a visible portion of the character, a leading width corresponding to the width of blank space preceding the visible portion of the character, and a trailing width corresponding to the width of blank space succeeding the visible portion of the character; and

for each character of the basis font, increasing or decreasing the leading width and trailing width, so that the overall width is equal to the overall width of the corresponding character of the unavailable font and so that the visual appearance of each character of the basis font is not changed

wherein when the requesting program displays characters using the basis font with the increased or decreased leading widths and trailing widths the characters are displayed with overall widths that are equal to the overall widths of the corresponding character of the unavailable font without further increasing or decreasing of the overall widths of the dismayed characters.

9. The method of claim 8, further including the step of resizing the characters of the basis font both horizontally and vertically to match a requested height.

10. The methods of claim 8 or 9, further including the step of regularizing the characters of the basis font.

11. The methods of claim 10, further including the step of converting each of the regularized characters of the basis font to a character bitmap.

12. The method of claim 9 in which the selecting step selects the candidate font for which the overall width of each character deviates the least from that of the corresponding character of the unavailable font.

13. The method of claim 8 in which the selecting step selects the most visually similar candidate font for which the overall width of each character deviates from the corresponding character of the unavailable font by less than a threshold proportion.

14. A method in a computer system for creating a substitute font for an unavailable font using available fonts whose characters each have an overall width that is the sum of an extent corresponding to the width of a visible portion of the character, a leading width corresponding to the width of blank space preceding the visible portion of the character, and a trailing width corresponding to the width of blank space succeeding the visible portion of the character, one or more of the available fonts being horizontally scalable, the method comprising the steps of:

receiving a request from a requesting program for a requested font;

determining that the requested font is unavailable in the computer system;

determining whether the overall width of each character of an available font selected to replace the requested font is within a predetermined tolerance of the overall width of the corresponding character of the unavailable font;

if it is determined that the overall width of each character of an available font is within the predetermined tolerance of the overall width of the corresponding character of the unavailable font, increasing or decreasing the leading and trailing widths of each character of that available font so that the overall width of each character of that available font is equal to the overall character width of the corresponding character of the unavailable font and so that the visual appearance of each character of the basis font is not changed and wherein when the requesting program displays characters using the basis font with the increased or decreased leading widths and trailing widths, the characters are displayed with overall widths that are equal to the overall widths of the corresponding character of the unavailable font without further increasing or decreasing of the overall width of the displayed characters; and

if it is determined that none of the available fonts has characters whose overall widths are all within the predetermined tolerance of the overall widths of the corresponding characters of the unavailable font, horizontally scaling the characters of a selected horizontally scalable font go that the overall width of each character of the selected font is equal to the overall width of the corresponding character of the unavailable font.

15. The method of claim 14, further including the step of selecting an available font that is visually similar to the unavailable font.

16. The method of claim 15, further including the step of regularizing the appearance of the horizontally scaled characters of the selected font.

17. The method of claim 16 wherein the characters of the selected font have vertical stems and in which the regularizing step includes the steps of:

identifying the character of the selected font whose overall width was changed the least by horizontal scaling; and

changing the widths of the vertical stems in every character of the selected font to the width of the vertical stems in the identified character.

18. The method of claim 15 in which the selected available font used in the determining step is similar in appearance to the unavailable font.

19. A method in a computer system for creating a substitute font for an unavailable font using available fonts whose characters each have an overall height that is the sum of an extent corresponding to the height of a visible portion of the character, an upper height corresponding to the height of blank space above the visible portion of the character, and a lower height corresponding to the width of blank space below the visible portion of the character, one or more of the available fonts being vertically scalable, the method comprising the steps of:

receiving a request from a requesting program for a requested font;

determining that the requested font is unavailable in the computers system;

determining whether the overall height of each character of a selected available font to replace the requested font is within a predetermined tolerance of the overall height of a corresponding character of the unavailable font, the selected available font having a character that corresponds to each of the characters of the requested font;

if it is determined that the overall height of each character of an available font is within the predetermined tolerance of the overall height of the corresponding character of the unavailable font, increasing or decreasing the upper and lower heights of each character of that available font so that the overall height of each character of that available font is the same as the overall height of the corresponding character of the unavailable font and so that the visual appearance of each character of the available font is not changed and wherein when the requesting program displays characters using the available font with the increased or decreased upper heights and lower heights, the characters are displayed with overall heights that are equal to the overall heights of the corresponding character of the unavailable font without further increasing or decreasing of the overall heights of the displayed characters; and

if it is determined that none of the available fonts has characters whose overall heights are all within the predetermined tolerance of the overall heights of the corresponding characters of the unavailable font, vertically scaling the characters of a selected vertically scalable font so that the overall height of each character of the selected font is the same as the overall height of the corresponding character of the unavailable font.

20. The method of claim 19, further including the step of selecting an available font that is visually similar to the unavailable font.

21. The method of claim 20, further including the step of regularizing the appearance of the vertically scaled characters of the selected font.

22. The method of claim 21 wherein the characters of the selected font have horizontal stems and in which the regularizing step includes the steps of:

identifying the character of the selected font whose overall height was changed the least by vertical scaling; and

changing the heights of the horizontal stems in every character of the selected font to the height of the horizontal stems in the identified character.

23. A method in a computer system for providing a substitute font for an unavailable font having characters with overall widths, the method comprising the steps of:

receiving a request from a requesting program for a requested font;

determining that the requested font is unavailable in the computer system;

selecting a font to replace the requested font having characters with vertical stems and overall widths that is available within the computer system, the vertical stems each having a width measured horizontally, the overall widths including a leading width, an extent width and a trailing width;

scaling the characters of the selected font horizontally to match the overall character widths of the unavailable font without adjusting the extent width of the characters so that the visual appearance of each character is not changed;

resizing the scaled characters of the selected font both horizontally and vertically to match a requested height; and

modifying the vertical stem widths of the resized characters of the selected font so that the vertical stem width of every resized character is the same

wherein when the requesting program displays characters using the selected font after its characters have been scaled, resized, and modified, the characters are displayed with overall widths that match the overall widths as if the characters were displayed in the requested font, resized to the requested height without further adjusting of the overall widths of the displayed characters.

24. The method of claim **23**, further including the step of converting the regularized characters of the substitute font each to a character bitmap.

25. The method of claim **23** in which the selecting step selects the selected font based on the classification of the unavailable font.

26. The method of claim **25** in which the selecting step selects a different selected font depending on whether the unavailable font is a serif font or a sans serif font.

27. A method in a computer system for creating a substitute font for an unavailable font using available fonts whose characters each have at overall width that is the sum of an extent corresponding to the width of a visible portion of the character, a leading width corresponding to the width of blank space preceding the visible portion of the character, and a trailing width corresponding to the width of blank space succeeding the visible portion of the character, one or more of the available fonts being horizontally scalable, the method comprising the steps of:

receiving a request from a requesting program for a requested font;

determining that the requested font is unavailable in the computer system;

identifying available fonts with appearances that are similar to the appearance of the unavailable font;

for each of the identified fonts, in the order of the identified fonts' similarity to the unavailable font, determining whether the overall width of each character of the identified font is within a predetermined tolerance of the overall width of the corresponding character of the unavailable font;

if it is determined that the overall width of each character of an identified font is within the predetermined tolerance of the overall width of the corresponding character of the unavailable font:

retrieving the identified font whose characters' overall widths were determined to be within a predetermined tolerance of the overall widths of the corresponding characters of the unavailable font, and

increasing or decreasing the leading and trailing widths of each character of the retrieved font so that the overall width of each character of the retrieved font is equal to the overall width of the corresponding character of the unavailable font and so that the visual appearance of each character of the retrieved font is not changed and wherein when the requesting program displays characters using the retrieved font with the increased or decreased leading widths and trailing widths, the characters are displayed with overall widths that are equal to the overall widths of the corresponding character of the unavailable font without further increasing or decreasing of the overall widths of the displayed characters; and

if it is determined that none of the available fonts has characters whose overall widths are all within the predetermined tolerance of the overall widths of the corresponding characters of the unavailable font:

selecting a horizontally scalable font,

retrieving the selected font,

horizontally scaling the characters of the selected font so that the overall width of each character of the selected font is equal to the overall width of the corresponding character of the unavailable font, and

regularizing the appearance of the horizontally scaled characters of the selected font.

28. The method of claim **27** in which the regularizing step includes the steps of:

identifying the character of the selected font whose overall width was changed the least by horizontal scaling; and

changing the widths of the vertical stems in every character of the selected font to the width of the vertical stems in the identified character.

29. An apparatus for providing a substitute font that visually approximates a requested font that is unavailable in the computer system, the requested font and the substitute font having respective sets of characters each of which have an overall width, the overall width having a leading width, an extent width, and a trailing width, the requested font and the substitute font being outline fonts, the apparatus comprising:

means for receiving a request from a requesting program for a requested font;

means for determining that the requested font is unavailable in the computer system;

means for selecting as the substitute font to replace the requested font a font that is available in the computer system;

means for adjusting the overall widths of the characters of the substitute font to match the overall widths of corresponding characters of the requested font by adjusting the leading and trailing widths without adjusting the extent width so that the visual appearance of each character of the substitute font is not changed; and

means for making the substitute font with the overall widths of the characters adjusted to match the overall widths of the requested font available to the requesting program so that when the requesting program displays characters using the substitute font and without further adjustment of the overall widths of the characters, the characters are displayed with overall widths that match the overall widths as if the characters had been displayed using the requested font.

30. The apparatus of claim **29**, further including means for adjusting predetermined features in the characters of the substitute font prior to making the substitute font available so that the predetermined features in the characters of the substitute font are consistent with each other.

31. The apparatus of claim **30** wherein one of the predetermined features in the characters of the substitute font is the widths of stems of the characters.

32. The apparatus of claim **31** wherein the stems that are operated upon by the adjusting means are the vertical stems of the characters of the substitute font.

33. The apparatus of claim **31** wherein the adjusting means adjusts widths of the stems of the substitute font characters to match the width of a stem of the substitute font character that has an overall width that is closest to the overall width of the corresponding selected font character so that the widths of the stems of the substituted fonts passed the substitute font to the program that has requested the selected font are equal.

34. The apparatus of claim **29** wherein the computer system contains a plurality of fonts, and wherein the apparatus further includes means for comparing the characters of each of the fonts in the computer system to the characters of the selected font, and selecting as the substitute font a font that is visually similar to the selected font.

35. An apparatus for providing a substitute font for an unavailable font having characters with overall widths, the apparatus comprising:

means for receiving a request from a requesting program for a requested font;

means for determining that the requested font is unavailable in the computer system;

means for identifying one or more candidate fonts that are available within the computer system to replace the unavailable font having compact numerical characterizations that differ from that of the unavailable font by less than a preselected maximum distance;

means for selecting a font from among the candidate fonts that has characters with overall widths, each of which is the sum of a leading width, an extent width, and a trailing width; and

means for increasing or decreasing the leading width and trailing width for each character of the selected font without adjusting the extent width, so that the overall width is equal to the overall width of the corresponding character of the unavailable font and so that the visual appearance of each character of the selected font is not changed

wherein when the requesting program displays characters using the selected font with the increased or decreased leading widths and trailing widths, the characters are displayed with overall widths that are equal to the overall width of the corresponding character of the unavailable font without further increasing or decreasing of the overall widths of the displayed characters.

36. The apparatus of claim **35**, further including means for resizing the characters of the selected font both horizontally and vertically to match a requested height.

37. The apparatus of claim **35** or **36**, further including means for regularizing the characters of the selected font.

38. The apparatus of claim **37**, further including a raster converter that converts means for the regularized characters of the selected font each to a character bitmap.

39. The apparatus of claim **35** in which the selecting means selects the most visually similar candidate font for which the overall width of each character deviates from the corresponding character of the unavailable font by less than a threshold proportion.

40. The apparatus of claim **39** in which the selecting means selects the candidate font for which the overall width of each character deviates the least from that of the corresponding character of the unavailable font.

41. An apparatus for creating a substitute font for an unavailable font using available fonts whose characters each have an overall width that is the sum of an extent width corresponding to the width of a visible portion of the character, a leading width corresponding to the width of blank space preceding the visible portion of the character, and a trailing width corresponding to the width of blank space succeeding the visible portion of the character, one or more of the available fonts being horizontally scalable, the apparatus comprising:

means for receiving a request from a requesting program for a requested font;

means for determining that the requested font is unavailable in the computer system;

means for determining whether the overall width of each character of a selected available font to replace the requested font is within a predetermined tolerance of the overall width of the corresponding character of the unavailable font;

means for increasing or decreasing the leading and trailing widths of each character of that available font so that the overall width of each character of that available font is equal to the overall character width of the corresponding character of the unavailable font if it is determined that the overall width of each character of an available font is within the predetermined tolerance of the overall width of the corresponding character of the unavailable font so that the visual appearance of each character of the available font is not changed and wherein when the requesting program displays characters using the available font with the increased or decreased leading widths and trailing widths, the characters are displayed with overall widths that are equal to the overall widths of the corresponding character of the unavailable font without further increasing or decreasing of the overall widths of the displayed characters; and

means for horizontally scaling the characters of a selected horizontally scalable font so that the overall width of each character of the selected font is equal to the overall width of the corresponding character of the unavailable font if it is determined that none of the available fonts has characters whose overall widths are all within the predetermined tolerance of the overall widths of the corresponding characters of the unavailable font.

42. The apparatus of claim **41**, further including means for selecting an available font that is visually similar to the unavailable font.

43. The apparatus of claim **42**, further including means for regularizing the appearance of the horizontally scaled characters of the selected font.

44. The apparatus of claim **43** wherein the characters of the selected font have vertical stems and in which the regularizing means includes:

means for identifying the character of the selected font whose overall width was changed the least by horizontal scaling; and

means for changing the widths of the vertical stems in every character of the selected font to the width of the vertical stems in the identified character.

45. The apparatus of claim **42** in which the selected available fonts considered by the determining means is similar in appearance to the unavailable font.

46. An apparatus for creating a substitute font for an unavailable font using available fonts whose characters each have an overall height that is the sum of all extent corresponding to the height of a visible portion of the character, an upper height corresponding to the height of blank space above the visible portion of the character and a lower height corresponding to the width of blank space below the visible portion of the character, one or snore of the available fonts being vertically scalable, the apparatus comprising:

means for receiving a request from a requesting program for a requested font;

means for determining that the requested font is unavailable in the computer system;

means for determining whether the overall height of each character of a selected available font to replace the requested font is within a predetermined tolerance of the overall height of the corresponding character of the unavailable font;

means for increasing or decreasing the upper and lower heights of each character of that available font so that the overall height of each character of that available font is equal to the overall character height of the corresponding character of the unavailable font if it is determined that the overall height of each character of an available font is within the predetermined tolerance of the overall height of the corresponding character of the unavailable font and so that the visual appearance of each character of the available font is not changed wherein when the requesting program displays characters using the available font with the increased or decreased upper heights and lower heights, the characters are displayed with overall heights that are equal to the overall heights of the corresponding character of the unavailable font without further increasing or decreasing of the overall heights of the displayed characters; and

means for vertically scaling the characters of a selected vertically scalable font so that the overall height of each character of the selected font is equal to the overall height of the corresponding character of the unavailable font if it is determined that none of the available fonts has characters whose overall heights are all within the predetermined tolerance of the overall heights of the corresponding characters of the unavailable font.

47. The apparatus of claim **46**, further including means for selecting an available font that is visually similar to the unavailable font.

48. The apparatus of claim **47**, further including means for regularizing the appearance of the vertically scaled characters of the selected font.

49. The apparatus of claim **48** wherein the characters of the selected font have horizontal stems and in which the regularizing means includes:

means for identifying the character of the selected font whose overall height was changed the least by vertical scaling; and

means for changing the heights of the horizontal stems in every character of the selected font to the height of the horizontal stems in the identified character.

50. A computer-readable medium containing instructions for causing a computer system to provide a substitute font that visually approximates a requested font that is unavailable in the computer system, the requested font and the substitute font having respective sets of characters each of which have an overall width, the overall width having a leading width, an extent width, and a trailing width, the requested font and the substitute font being outline fonts, by:

receiving a request for a requested font; and

when the requested font is unavailable,

selecting as the substitute font to replace the requested font a font that is available;

adjusting the overall widths of the characters of the substitute font to match the overall widths of corresponding characters of the requested font by adjusting the leading and trailing widths so that the visual appearance off each character of the substitute font is not changed; and

making the substitute font with the overall widths of the characters adjusted to match the overall widths of the requested font available for use in place of the requested font so that when the requesting program displays characters using the substitute font and without further adjustment of the overall widths of the characters, the characters are displayed with overall widths that match the overall widths as if the characters had been displayed using the requested font.

51. The computer-readable medium of claim **50**, further including adjusting predetermined features in the characters of the substitute font prior to making the substitute font available so that the predetermined features in the characters of the substitute font are consistent with each other.

52. The computer-readable medium of claim **51** wherein one of the predetermined features in the characters of the substitute font is the width of stems of the characters.

53. The computer-readable medium of claim **52** wherein the stems that are adjusted are the vertical stems of the characters of the substitute font.

54. The computer-readable medium of claim **52** wherein the widths of the stems of the substitute font characters are adjusted to match the width of a stem of the substitute font character that has an overall width that is closest to the overall width of the corresponding selected font character so that the widths of the stems of the substituted fonts passed the substitute font to the program that has requested the selected font are equal.

55. The computer-readable medium of claim **50** wherein the computer system contains a plurality of fonts, and further including comparing the characters of each of the fonts in the computer system to the characters of the selected font, and selecting as the substitute font a font that is visually similar to the selected font.

56. A computer-readable medium containing instructions for causing a computer system to provide a substitute font for an unavailable font having characters with overall widths, the unavailable font having a numerical characterization of its visual characteristics, by:

receiving a request for a requested font; and

when the requested font is unavailable,

identifying one or more candidate fonts that are available to replace the unavailable font having numerical characterizations of their visual characteristics that differ from that of the unavailable font by less than a preselected maximum distance;

selecting a basis font from among the candidate fonts that has characters with overall widths, each of which is the

sum of an extent width corresponding to the width of a visible portion of the character, a leading width corresponding to the width of blank space preceding the visible portion of the character, and a trailing width corresponding to the width of blank space succeeding the visible portion of the character; and

for each character of the basis font, increasing or decreasing the leading width and trailing width, so that the overall width is equal to the overall width of the corresponding character of the unavailable font and so that the visual appearance of each character of the basis font is not changed

wherein when the requesting program displays characters using the basis font with the increased or decreased leading widths and trailing widths the characters are displayed with overall widths that are equal to the overall widths of the corresponding character of the unavailable font without further increasing or decreasing of the overall widths of the displayed characters.

57. The computer-readable medium of claim **56**, further including resizing the characters of the basis font both horizontally and vertically to match a requested size.

58. The computer-readable medium of claim **56**, further including the step of regularizing the characters of the basis font.

59. The computer-readable medium of claim **58**, further including converting each of the regularized characters of the basis font to a character bitmap.

60. The computer-readable medium of claim **56** in which the selecting of a basis font selects the candidate font for which the overall width of each character deviates the least from that of the corresponding character of the unavailable font.

61. The computer-readable medium of claim **56** in which the selecting of a basis font selects the most visually similar candidate font for which the overall width of each character deviates from the corresponding character of the unavailable font by less than a threshold proportion.

* * * * *